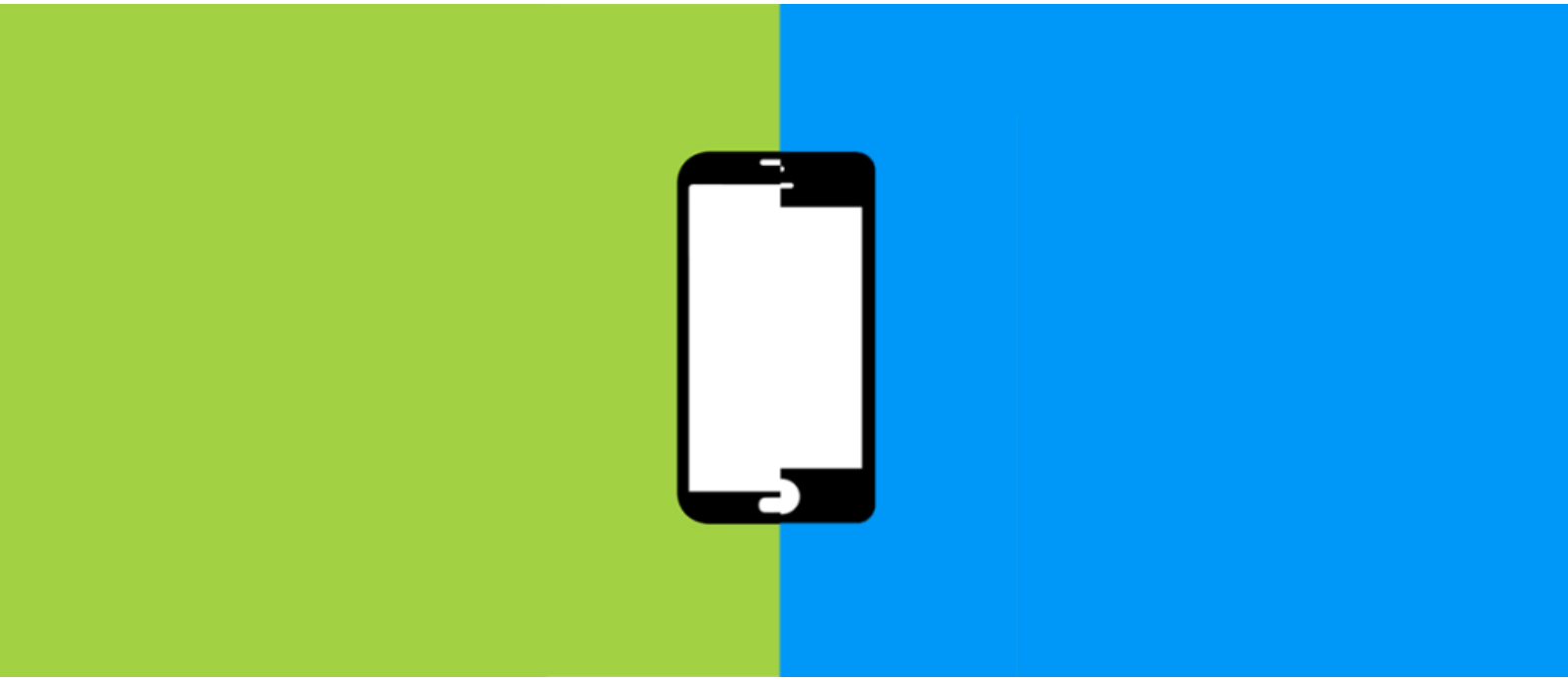
TRABAJO FINAL DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

ENGINYERIA DEL SOFTWARE



DESARROLLO DE UNA APP MULTIPLATAFORMA PARA LA RESERVA Y GESTIÓN DE SALAS EN FAMA/AFM

Autor
David Marin Medina

Ponente
Albert Cabellos Aparicio

Director
Valentí Creus Vall

Universitat Politécnica de Catalunya

2018

Contenido

1. Introducción 5
2. Formulación del problema y contextualización 6
2.1. Contextualización 6
2.2. Formulación del problema 6
2.3. Stakeholders 7
3. Fase Inicial 7
3.1. Alcance 8
3.2. Marco contextual 8
3.2.1. Estado del arte 8
3.2.2. Obstáculos y riesgos 12
3.3. Desarrollo y medios 12
3.3.1. Metodología 12
3.3.2. Herramientas de desarrollo 14
3.3.3. Herramientas de seguimiento 14
3.3.4. Prototipaje 15
3.4. Planificación temporal 15
3.4.1. Fases del proyecto 15
3.4.2. Recursos 16
3.4.3. Planificación temporal 17
3.4.4. Alternativas y plan de acción 20
3.5. Presupuesto 22
3.5.1. Costes directos 22
3.5.2. Costes indirectos 24
3.5.3. Costes inesperados 25
3.5.4. Costes de contingencia 25
3.5.5. Coste total y gestión de control 25
3.6. Sostenibilidad y Compromiso social 26
3.6.1. Dimensión económica 26
3.6.2. Dimensión ambiental 27
3.6.3. Dimensión social 27
3.6.4. Matriz de sostenibilidad 28
4. Análisis funcional y especificación de los requisitos 30
4.1. Requisitos de la App 30

4.2. Especificación funcional 31
4.2.1. Casos de uso 31
4.2.2. Diagrama de casos de uso 32
4.3. Especificación no funcional 34
4.3.1. Requisitos no funcionales 34
4.3.2. Requisitos técnicos 35
4.4. Cambios en el FAMA/AFM 35
4.4.1. Cambios en el modelo de datos 36
4.4.2. Casos de uso de la aplicación Web 36
4.4.3. Servicios web 37
4.5. Esquema de la APP 39
4.5.1. Patrón modelo-vista-modelo de vista 39
4.5.2. Estructura interna 43
4.6. Diseño de pantallas 44
4.6.1. Pantalla *login* y pantalla principal 44
4.6.2. Reserva actual 45
4.6.3. Reserva directa 46
4.6.4. Reserva mediante buscador 47
4.6.5. Historial de reservas 50
5. Desarrollo 52
5.1. Plan de Acción 52
5.2. Desarrollo 53
5.3. Incidencias y contratiempos 63
6. Plan de pruebas 69
7. Publicación 72
7.1. Android y Google Play 73
7.1.1. Compilación 74
7.1.2. Publicación 75
7.2. iOS y App Store 76
7.2.1. Compilación 76
7.2.2. Publicación 77
8. Futuras versiones 79
8.1. Resolución de Incidencias 79
8.2. Mejoras 80
8.3. Nuevas funcionalidades 81

9. Aspecto final 83
9.1. Pantallas 83
9.1.1. Inicio de sesión 83
9.1.2. Menú principal 84
9.1.3. Acerca de 86
9.1.4. Reserva de Sala 87
9.1.5. Buscador 89
9.1.6. Lista resultado Buscador 91
9.1.7. Histórico 92
9.2. Mapa de clases 93
10. Conclusiones 95
11. Biografía 97
12. Anexo 100
12.1. Códigos externos 100
ACR User Dialogs 100
Newtonsoft.Json 100
Permissions Plungin y Geolocator Plungin 101
Media Plungin 101
12.2. Página de *Google Play* de la Aplicación 102
12.3. Página de *App Store* de la Aplicación 103

1. Introducción

Dentro del mundo de la información en el que nos movemos diariamente cada vez queda más expuesta la **necesidad de administrar y gestionar** todas esas corrientes de datos que envuelven los distintos elementos de nuestra vida y más en concreto, en el ámbito laboral.

En los entornos profesionales, esta gestión de datos puede ser mayor o menor, más o menos relevante y más o menos usada entre los distintos trabajadores de un entorno en concreto, pero **siempre se encuentra presente** de alguna u otra forma.

Hondando más en el tema, estos datos a gestionar pueden ir desde "sencillos" registros de inmuebles de una empresa por ejemplo hasta complejos ciclos de trabajo en los que pueden llegar a intervenir un número importante de personas.

En el ámbito laboral, la disciplina encargada de definir y estudiar esta gestión es el **Facility Management**.

El **Facility Management** es el instrumento que asegura el buen funcionamiento de las infraestructuras de las organizaciones y de sus servicios asociados de la manera más eficiente y óptima, mediante la integración de personas, espacios, procesos y las tecnologías. La normativa Europea en Facility Management 15221/1, define el Facility Management como "la gestión de inmuebles y servicios de soporte".

"Según los cálculos de IFMA (Asociación Internacional de Facility Management)*, esta actividad representa el 30% de los gastos de una empresa, gastos […] relacionados con el correcto funcionamiento, conservación y mantenimiento de las instalaciones en las que se ubica una compañía. Según IFMA, el […] facility manager puede conseguir reducir esta factura entre un 20% y un 30"* (El Confidencial, Elena Sanz, 2014)[1]

Volviendo a la administración de datos, esta no siempre es gestionada por personas que se dediquen a ello de manera en específica, es decir, esta gestión no siempre será la tarea principal a desarrollar y se debe realizar de la manera más eficiente posible. Para ello, una de las consideraciones más importantes a tener en cuenta es el uso de las **herramientas adecuadas** que permitan dar una respuesta integral y afectiva a los diferentes usuarios, resultando así una mejor gestión de una empresa, que se acabará traduciendo en la reducción y optimización de gastos.

En este punto nos encontramos con el motivo de este Trabajo de Final de Grado: El desarrollo de una aplicación que facilite una tarea de gestión concreta a los trabajadores de una empresa de una manera sencilla.

[1] "El 'facility manager', el gran desconocido que puede ahorrar millones a una empresa" https://www.elconfidencial.com/vivienda/2014-03-14/el-facility-manager-el-gran-desconocido-que-puede-ahorrar-millones-a-una-empresa_96490/

2. Formulación del problema y contextualización

En esta sección desgranaremos un poco más los conceptos básicos y las motivaciones de este trabajo, así como las consideraciones tomadas que nos llevan a la solución en forma de App y no en otro formato.

2.1. Contextualización

Para poder empezar a entrar en materia y definir este proyecto, primero tenemos que conocer otro software en el que se apoyará esta aplicación: **FAMA AFM.**

Tal como lo define la propia empresa, FAMA AFM es la solución integral de Facility Management, tipo software CAFM (Computer Aided Facility Management) e IWMS (Integrated Workplace Management System) desarrollado por la empresa Fama Systems, especializado en dar esa respuesta integral y efectiva a las organizaciones[2].

Más en concreto, es un software que permite llevar a cabo un exhaustivo control y una gestión integral de los diferentes aspectos que contribuyen al buen funcionamiento de las infraestructuras de las organizaciones y de sus servicios asociados.

En este entorno, una empresa puede registrar y gestionar todos sus inmuebles, empleados, flujos de trabajo, e incluso gestionar aspectos tan concretos como la gestión de contratos, impacto medioambiental o la administración de espacios y puestos de trabajo y es aquí donde se ubica este proyecto.

2.2. Formulación del problema

El software de Fama Systems es accesible desde el Portal Fama, a través de un explorador web. En este punto surgen dos inconvenientes: **Se quiere desarrollar una solución que permita gestionar la reserva de las diferentes salas y/o espacios de una empresa y su debida administración,** siendo este el primer inconveniente, pues la funcionalidad no existe dentro del sistema.

El segundo de los inconvenientes lo tenemos al ver quién hará uso de dicha funcionalidad y observar que la mayoría de usuarios no tiene un ordenador a mano la mayor parte del tiempo y acceder a este a través de un dispositivo móvil no daría al usuario la experiencia deseada.

Partiendo de este análisis básico se propone buscar una alternativa al posible desarrollo de un módulo adicional en FAMA AFM para esta funcionalidad, poniendo como prioridad la accesibilidad.

Llegados a este punto y poniendo en el eje central los dispositivos móviles, la propuesta más llamativa es la confección de una app debido a la facilidad y la versatilidad que ofrecen los mismos.

[2] "FAMA AFM, Software de Facility Management integral – Presentación" http://www.fama-systems.com/fama-afm-p-1-es

Más concretamente, la solución planteada constará en el desarrollo de una **aplicación multiplataforma** para solventar el problema de la accesibilidad y que implemente las funciones deseadas en comunicación con el sistema FAMA AFM.

2.3. Stakeholders

En este apartado identificaremos y definiremos los principales actores y partes interesadas, toda aquella persona u organización que tenga algún tipo de relación o interés sobre el proyecto, trabaje o no en él.

Desarrolladores

El equipo de desarrolladores del proyecto está formado personas que desempeñarán roles muy marcados y distinguidos, aunque en algún caso se podrían compartir. Los roles serán: Un desarrollador de aplicaciones, un diseñador, un ingeniero de requisitos (analista) y un gestor de servicios back-end y base de datos.

Director

El director del proyecto es Valentí Creus, subdirector de Fama Systems y será el principal encargado de la supervisión del proyecto, hacer el seguimiento y monitorizar tanto el desarrollo como el TFG.

Así mismo, también será quien se encargue de asesorar al estudiante en todo aquello relativo al desarrollo de las competencias técnicas y transversales asociadas al TFG.

Ponente

El ponente del trabajo por parte de la facultad es Albert Cabellos, el cual desempeñará un papel de acompañamiento al estudiante durante el trabajo y se encargará de la supervisión y asesoramiento para garantizar el desarrollo del trabajo dentro de los plazos establecidos y con la calidad y garantías esperadas en un TFG.

Administrador de sistema

Por la parte de cliente nos encontramos la persona encargada de administrar el sistema. Esta persona será la que, en cada contexto diferente en la que se quiera usar el sistema, registre los inmuebles y sus respectivas salas y espacios para que el usuario final pueda realizar las solicitudes de reservas con la App.

Usuario

Finalmente, el usuario final será aquella persona que, dentro de su ámbito laboral, use la aplicación para realizar una reserva de una sala o espacio dentro de uno de los inmuebles de una empresa.

Este proyecto va dirigido a todos los usuarios de las empresas clientes de Fama Systems S.A. que usarán la aplicación de gestión y reserva de salas integrada con el software FAMA AFM.

3. Fase Inicial

Una vez identificado el problema y partiendo del contexto descrito anteriormente, realizaremos los pasos previos al desarrollo: acotaremos el alcance,

estudiaremos las herramientas y medios que se usarán durante el proyecto, daremos un paso más en el estudio de contexto y exploraremos soluciones existentes que puedan resultar similares al proyecto planteado en este trabajo. Finalmente definiremos una planificación e identificaremos los recursos a invertir en el mismo.

3.1. Alcance

Teniendo un concepto más completo y analizado, veremos a continuación el alcance del proyecto y los distintos grandes objetivos y retos que afrontar durante el desarrollo de la aplicación:

Preparación del entorno FAMA AFM

Debido a que tratamos una funcionalidad nueva dentro del entorno FAMA AFM nos encontramos con que el sistema no implementa esta funcionalidad.

Antes de empezar a diseñar la aplicación hay que estudiar el modelo de datos de la aplicación, ver como se integran las salas y espacios y como ampliar el modelo y demás aspectos de la aplicación para la interacción de la aplicación.

Creación de la App multiplataforma

Como se ha comentado en el planteamiento inicial, en la formulación del problema, FAMA AFM es un software que se accede y trabaja vía web y no está pensado para la navegabilidad en una aplicación móvil aunque sea técnicamente posible. La aplicación se tiene que diseñar desde cero y multiplataforma, ideando una solución que se pueda usar en un sistema Android como en un sistema IOS.

Es necesario remarcar que el tiempo para el desarrollo de dos App es limitado y será necesario el uso de un framework integrador que permita el desarrollo simultáneo multiplataforma.

Integración en la API de FAMA

A parte de la preparación del modelo de datos y los aspectos varios de los que partirá el desarrollo como base, la aplicación hará un gran uso de la API de FAMA mediante servicios SOAP o REST. Este será un aspecto obligatorio y será imperativo el uso de un framework que se integre adecuadamente con estas arquitecturas y protocolos.

3.2. Marco contextual

3.2.1. Estado del arte

En este apartado hablaremos del estado de las tecnologías de gestión de datos, frameworks de desarrollo de aplicaciones y más en concreto de aplicaciones de reserva y gestión de espacios, así como aplicaciones que puedan recoger varios de los aspectos que se quieran desarrollar en la App.

También evaluaremos y confirmaremos si el planteamiento inicial del proyecto ha sido el correcto con la elección de una app como forma a la funcionalidad a desarrollar.

Estudio de mercado

Debido a antiguos desarrollos de aplicaciones dentro de la empresa, el framework de trabajo va a resultar la combinación de **Visual Studio** con **Xamarin** al frente, pero no sin antes hacer un parón crítico en el estado de este entorno y su salud a nivel técnico en un mercado donde el desarrollo multiplataforma es tan volátil[3].

Fundado en 2011, Xamarin es una plataforma para desarrollar aplicaciones para plataformas iOS, Android , Windows Phone , Windows Store y Mac usando el lenguaje de programación C# con múltiples guiños al desarrollo de aplicaciones en .NET.

Actualmente, los puntos fuertes de esta plataforma no han decaído; Ofrece una misma tecnología para el desarrollo tanto en Android e IOS, un rendimiento cercano al nativo, experiencias finales cercanas a las respectivas tanto visualmente como en funcionalidades y una tecnología de código abierto con un fuerte soporte por parte de Microsoft como aspectos a destacar.[4]

		Xamarin	React Native	Ionic
Running the code	iOS	AOT	Interpreter	Interpreter JIT with plugin
	Android	JIT/AOT	JIT	JIT
64-bit support	iOS	YES	?	Yes
	Android	Yes	No	?
GUI	iOS and Android	Native widgets (directly or under the hood)	Native widgets under the hood	HTML

	Xamarin	React Native	Ionic
Performance	Good	64-bit issues Interpreter	HTML
Native look	Yes	No	Yes
Modern development features	Classic only	Hot reloading Instant updates	Live reloading Instant updates
OS features	Full support	Plugins with different interface, something is missing	Plugins with different interface, something is missing

Tabla 1: Comparación en diferentes aspectos Entre Xamarin, React Native e Iconic[5]

Podemos afirmar pues que Xamarin sigue siendo una apuesta fuerte para el desarrollo multiplataforma frente a otras opciones del mercado sin aspectos en contra que nos puedan afectar a nuestro proyecto[6][7] con un gran apoyo en las distintas comunidades de desarrolladores en la red.[8]

Software de gestión y reservas

En un mundo donde la informatización de los medios cada vez es más importante y eficiente, el software de gestión o de oferta de servicios es cada vez más amplio. Más en concreto, si hablamos de gestión y reservas podemos encontrar de diferente índole directamente en nuestro entorno en lugares como bibliotecas, alquiler de vehículos, o incluso restaurantes.

[3] "Cross-platform Frameworks for Mobile Development" https://medium.com/@MasterOfCodeGlobal/best-10-android-frameworks-for-building-android-apps-d2d0ee48e464
[4] "The Good and The Bad of Xamarin Mobile Development" https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/
[5] "Xamarin vs Ionic vs React Native: differences under the hood - Cruxlab." https://cruxlab.com/blog/reactnative-vs-xamarin/
[6] "Ionic vs. Xamarin vs. PhoneGap" https://stackshare.io/stackups/ionic-vs-phonegap-vs-xamarin
[7] "The Good and The Bad of Xamarin Mobile Development" - Xamarin Cons to Consider https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/
[8] "Ionic vs. Xamarin vs. PhoneGap" https://stackshare.io/stackups/ionic-vs-phonegap-vs-xamarin

Para un estudio específico de las distintas funcionalidades de un software de gestión y reservas de espacios funcional, cogí como referencia el *software de reserva de salas de la red de bibliotecas de la UPC*.

Aunque pueda resultar un software sencillo en apariencia, realiza de forma eficaz todas las funcionalidades esperadas en la reserva de salas y gestión de reservas. De este aplicativo se pueden identificar las siguientes funcionalidades:

- Identificación con un usuario personal para ligar la reserva.
- Muestra de todos los edificios con espacios disponibles para reservar.
- Lista de todas las salas disponibles de todos los edificios.
- Visionado de horarios de reserva con su disponibilidad.
- Vista previa de la reserva para su revisión.
- Lista de las reservas del usuario.

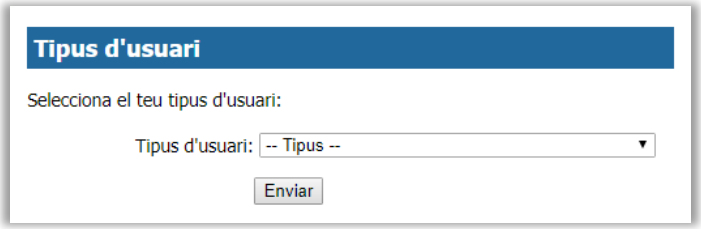


Figura 1.1: Pantalla de login con un usuario del sistema

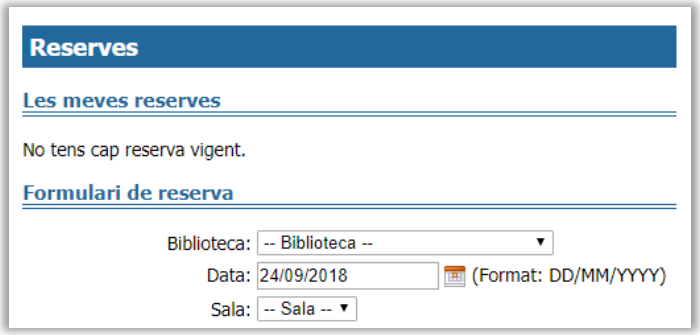


Figura 1.2: Pantalla de selección de edificio

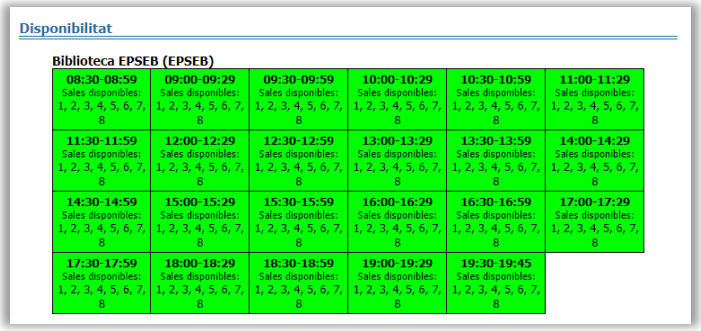


Figura 1.3: Pantalla con la lista de salas y su disponibilidad del edificio seleccionado

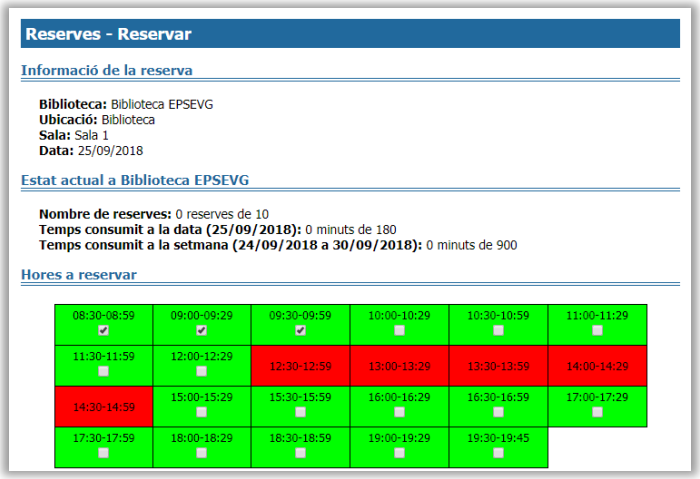


Figura 1.4: Pantalla de selección de horas a reservar en modo edición

Figura 1: Diferetnes pantallas del software de reserva de salas del conjunto de bibliotecass de la UPC

Este pequeño estudio nos da una idea de las funcionalidades que se consideran básicas y aunque ya se hayan tenido en cuenta en el planteamiento de las funcionalidades de la aplicación del proyecto nos ayuda a reafirmar que programas con estructuras funcionales similares son usadas de manera activa.

Aplicaciones móviles

El número de usuarios de móviles en el mundo alcanzó los 5.000 millones al finalizar 2017. Esto significa que el 66% de la población mundial cuenta con un móvil y lo usa habitualmente.

Los usuarios con acceso a internet llegan a los 4.021 mil millones (una penetración del 53%).[9] A demás, el móvil es el dispositivo más utilizado en España para acceder a internet, usado ya por el 97% de los españoles mediante Android y el 11% mediante iOS.[9]

Los usuarios de móviles pasan de media dos horas al día trasteando con aplicaciones […] las apps han cambiado los hábitos de miles de millones de usuarios.[10]

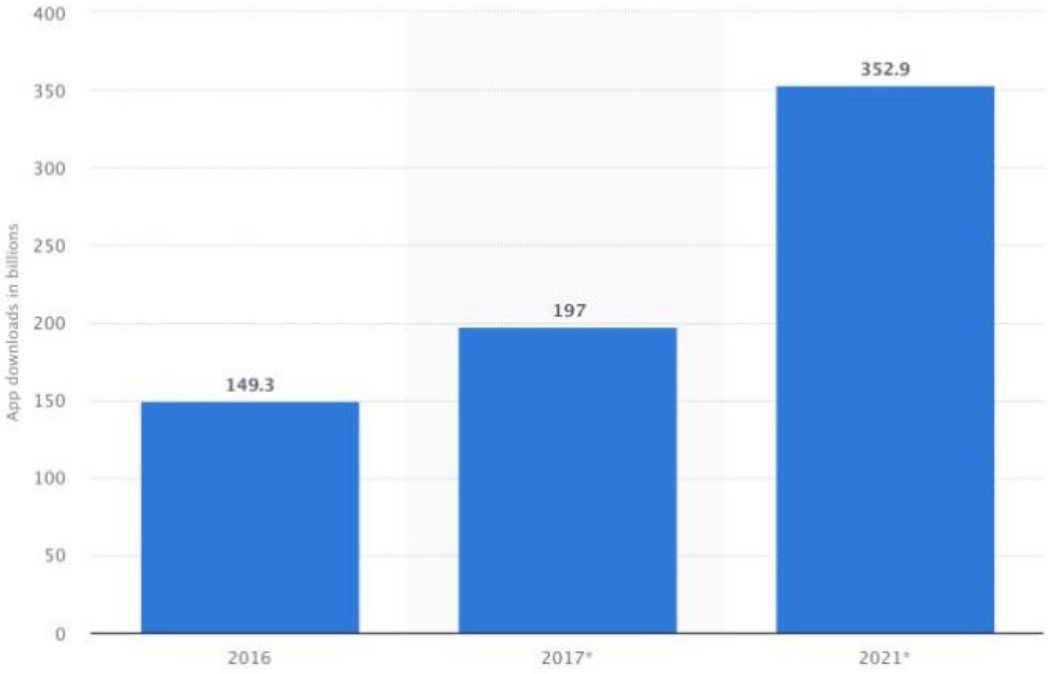


Figura 2: Número de descargas de aplicaciones móviles en todo el mundo en 2016, 2017 y 2021 (en miles de millones)

Desde PickASO nos muestran como con el paso de los años, el móvil se ha posicionado como un elemento imprescindible para realizar compras a través de apps desde la consulta de precios, información del producto hasta reviews.[11]

Todos estos datos nos arrojan un escenario donde las aplicaciones forman el nexo entre el día a día de las personas y los quehaceres de la vida cotidiana en el mundo virtual.

De una forma contundente podemos afirmar que, el mundo de las aplicaciones móviles es presente y futuro dentro del ámbito profesional para acercarse a los usuarios y facilitar el acceso a los diferentes servicios que estos puedan ofrecer.

[9] "Informe ditrendia: Mobile en España y en el Mundo 2018" https://ditrendia.es/informe-mobile-2018/
[10] "La explosión de la economia de las aplicacions" http://www.expansion.com/economia-digital/companias/2017/03/03/58ac1f92468aeb76208b458b.html
[11] "Predicciones sobre la Economía de las Apps" https://pickaso.com/2017/2018-predicciones-economia-apps

Maximizar estas posibilidades no puede más que beneficiar tanto al usuario como a las empresas que.

3.2.2. Obstáculos y riesgos

Una desventaja desde la cual parto al realizar este trabajo es el desconocimiento previo de las herramientas con las que se va a desarrollar la App.

Antes de empezar el desarrollo de la aplicación tendré que familiarizarme con el entorno y el framework de *Xamarin* y la evaluación de riesgos previos al desarrollo podría variar contando que podrían surgir problemas fruto del desconocimiento del entorno.

Posteriormente, aunque Xamarin es un framework estable que traduce las aplicaciones de forma nativa, el hecho de realzar una aplicación multiplataforma podría conllevar problemas más específicos de cada entorno, lo que añade un grado de complejidad e incertidumbre pues las soluciones podrían requerir de conocimientos en Android e IOS.

Del punto anterior nacen ciertos riesgos, como la visualización de los componentes y pantallas en las distintas plataformas o un comportamiento incorrecto por parte de *Xamarin.Forms,* los cuales podrían retrasar u obligar a realizar un cambio de rumbo en el diseño y el desarrollo de la aplicación.

Cabe destacar que, muchos de los riesgos son conocidos y son fácilmente identificables gracias a dos factores en los que se basará el futuro desarrollo: Uno será el desarrollo anterior a esta aplicación dentro de la empresa, una aplicación para gestionar Solicitudes de trabajo, que dio los primeros pasos con *Xamarin* y el desarrollo multiplataforma y el otro, y más importante, la comunidad de desarrolladores de *Xamarin*, con un gran bagaje y una gran documentación que conformará una gran fuente de consulta para la resolución de los riesgos mencionados.

3.3. Desarrollo y medios

En este apartado se describirá la forma en la que se llevará a cabo el desarrollo del trabajo, la metodología y las herramientas de desarrollo.

3.3.1. Metodología

Para poder llevar a cabo el proyecto de una manera organizada y asegurar su éxito, se ha definido una metodología de trabajo a seguir. Esta consta de una planificación dividida en 4 fases en las cuales englobarán diferentes tareas.

De las cuatro fases, la segunda y la tercera se realizarán de manera cíclica e incremental, de manera que podamos avanzar en el desarrollo de los diferentes aspectos de la aplicación después de haber recibido un feedback y haber hecho la evaluación pertinente al trabajo anterior.

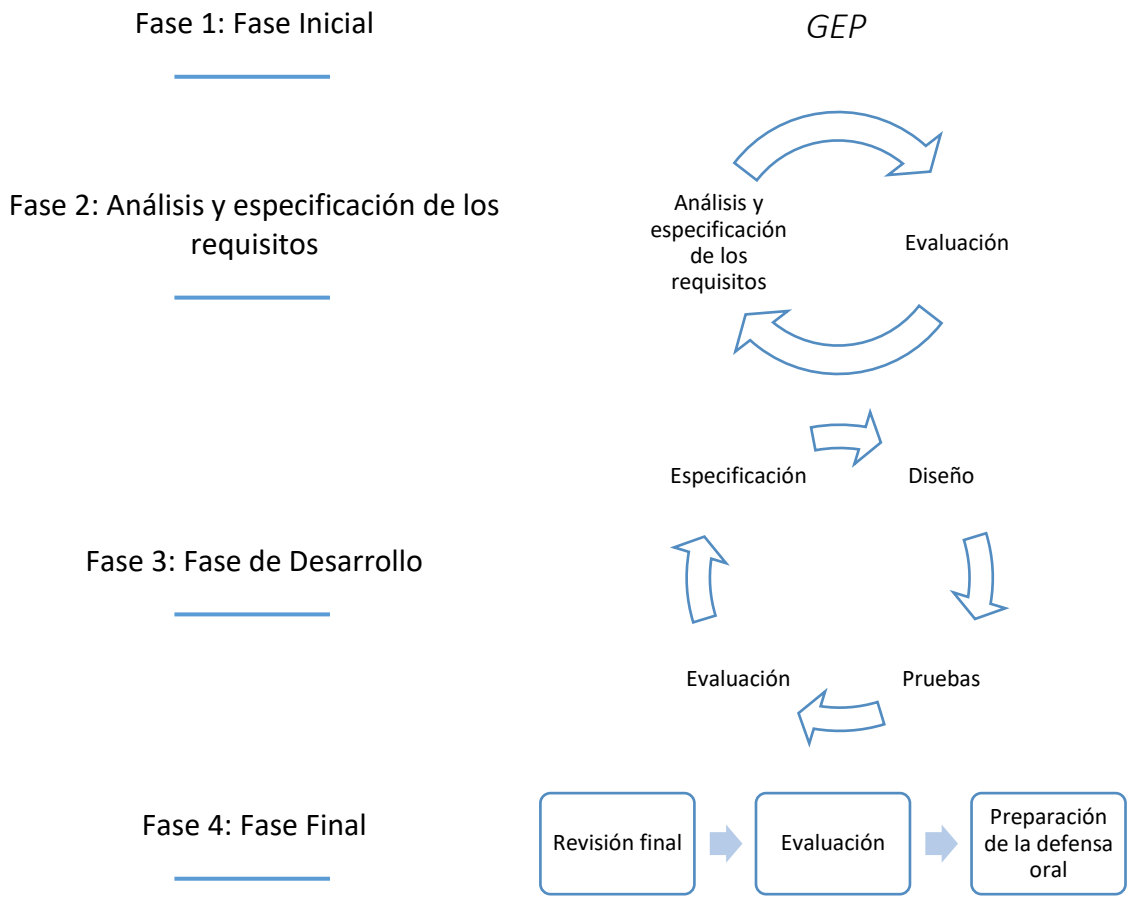


Figura 3: Detalle gráfico de las diferentes fases de la metodología de trabajo

Ciclo de desarrollo

Los ciclos de desarrollo son muy importantes por el hecho de poder realizar una evaluación continua del trabajo realizado, así como una puesta en común del conocimiento y del estado de las distintas partes del proyecto. Estos nos permiten evaluar detectar y corregir los distintos problemas que puedan surgir durante la implementación, así como tomar nuevos rumbos en el momento preciso si se considerase oportuno y necesario.

Normalmente un ciclo de trabajo se formará alrededor de una funcionalidad o partes cruciales dentro de esta, como es el ejemplo de las llamadas a servidor que pueden llegar a ser muy complejas.

Feedback de desarrollo

Paralelamente a la evaluación sobre las funcionalidades, la aplicación será probada por agentes externos al desarrollo, pudiendo aportar así una opinión externalizada y más global.

Estas personas se encargarán de poner a prueba la aplicación no solo en el ámbito que se está desarrollando en ese momento y sobre todo jugando el papel del cliente en cuanto a funcionalidad y evaluación.

3.3.2. Herramientas de desarrollo

Las herramientas de desarrollo serán todas las listadas a continuación:

Visual Studio	Visual Studio es el entorno principal de desarrollo. Junto a Xamarín, en él se desarrollará toda la aplicación tanto en Android como en IOS.
Xamarin	Herramienta integrada en Visual Studio que permite desarrollar una Aplicación móvil para distintas plataformas escribiéndola en un solo idioma.
Eclipse	Eclipse es otro entorno de desarrollo en el que se realizarán las llamadas a servidor por la parte del entorno FAMA AFM.
PLSQL Developer	Con PLSQL Developer tendremos un acceso directo a Base de datos para observar su comportamiento y modificar datos de cara a realizar pruebas.
iMac Pro	Para la compilación en IOS es necesario una Máquina con una licencia con IOS. El entorno en Xamarin se montará en esta máquina.
Nexus 4	La App Android se testeará En un Nexus 4 con Android 5.1 aunque el target mínimo es Android 4.4 .
Iphone 7	La App IOS se testeará en un Iphone 7 con la última versión de IOS.

3.3.3. Herramientas de seguimiento

VssConnect y SVN	Mediante el SVN de la empresa podremos realizar un control de versiones. Se añadirá un comentario a cada cambio efectuado en el código tras un *commit*.
Programa de horas de la empresa	Aunque no sea una herramienta de desarrollo en sí, el programa de horas de la empresa obliga a desgranar las horas invertidas en el proyecto y a realizar un control del tiempo invertido en los diferentes aspectos del proyecto.

3.3.4. Prototipaje

En algún momento temprano del desarrollo, cuando ya se asuman ciertos conocimientos en el entorno de programación, Xamarin, se generará una primera versión prototipo que servirá para analizar el desarrollo posterior y la planificación por si fuese necesario realizar modificaciones o reevaluar la dedicación de horas en los diferentes aspectos del proyecto.

Este primer prototipo también servirá para realizar una primera evaluación de los conocimientos adquiridos hasta el momento.

3.4. Planificación temporal

En los apartados anteriores pudimos ver expuesto el alcance del proyecto y el contexto del trabajo a realizar. En este apartado recuperaremos las fases del proyecto, intuiremos y asignaremos diferentes tareas a las mismas y las situaremos temporalmente según la planificación y acorde a la metodología.

Así mismo, tendremos que tener en cuenta que la planificación siempre pude verse alterada y aunque siempre se puede minimizar el impacto de los imprevistos mediante un ejercicio de prevención, esta no será inamovible y deberá permitir cierta flexibilidad, así como una evaluación posterior a todo el proceso.

3.4.1. Fases del proyecto

En esta parte describiremos y detallaremos las distintas fases principales de la realización del proyecto. Esta descripción concisa nos ayudará a entrever e identificar las distintas tareas dentro de las fases, así como una visión de los tiempos de cada una de ellas y el peso dentro del desarrollo.

Fase inicial

Esta primera fase la conforma principalmente la asignatura de Gestión de Proyectos (GEP). Como la propia asignatura de la facultad se presenta, esta tiene dos grandes objetivos: presentar las metodologías y herramientas que permitan desarrollar el TFG con éxito y reunir una gran base de material bibliográfico que posibilite desarrollar las competencias y habilidades que luego se requerirán durante el ejercicio del proyecto.[12]

Análisis y especificación de los requisitos

La segunda fase consta del bloque formado por la fase de análisis y especificación de requisitos. En esta fase se tendrá que definir la arquitectura del sistema a construir: el modelo de arquitectura a alto nivel, el modelo de datos y procesos en back-end, el modelo de sistemas y la organización durante el desarrollo.

[12] "GESTIÓ DE PROJECTES (GEP) - Guia de l'assignatura. Curs 2018-19 Q1" (Acceso restringido) https://atenea.upc.edu/pluginfile.php/2534788/mod_resource/content/1/Guia%20GEP%20Curs%202018-19%20Q1.pdf

Fase de desarrollo

La tercera fase es la que comprende todo el desarrollo del proyecto. La forman la documentación, el plan de acción y la implementación de todo el sistema especificado en las fases anteriores.

Fase final

La cuarta y última fase es la que reúne todo el trabajo realizado y engloba todo el proyecto. Estará comprendida por el despliegue del sistema, evaluación final y la preparación de la defensa oral del proyecto.

3.4.2. Recursos

En este apartado hablaremos de los recursos que se usarán en el proyecto. Estos recursos se pueden dividir entre los recursos de hardware y los recursos de software.

Recursos de Hardware

PC *(Intel Core i7, NVIDIA NVS 4200M, SSD de 500 GB, 8 GB RAM)*	Usado para el desarrollo de la ampliación del módulo del software FAMA7AFM y su portal y la ampliación de la Base de Datos para los datos persistidos de las reservas de salas e inmuebles.
iMac Pro *(Intel Core Xeon, GPU AMD Radeon Vega con 16 Gbytes de VRAM, SDD de 4 TB, 32 GB RAM)*	Usado para el desarrollo de la aplicación y la firma y distribución de las aplicaciones para iOS.
Android LG Nexus 4	Usado para las pruebas de la aplicación en el entorno Android de forma nativa.
iPhone 78	Usado para las pruebas de la aplicación en el entorno iOS de forma nativa.
Servidor web	Servidor físico que almacena los datos del software de la empresa contra el que la aplicación realizará las peticiones de datos pertinentes.

Recursos de Software

Windows 10	Usado para el desarrollo de la ampliación del módulo del software FAMA7AFM y su portal y la ampliación de la Base de Datos para los datos persistidos de las reservas de salas e inmuebles.
MAC OS Sierra	Usado para el desarrollo de la aplicación y la firma y distribución de las aplicaciones para iOS.

Herramientas de Office	Usados para la generación de la documentación requerida en el proyecto. Será necesaria una herramienta de procesado de Documentos, de presentaciones y de hojas de cálculo.
PLSQL Developer	Usado para el desarrollo de la ampliación de la Base de Datos para los datos persistidos de las reservas de salas e inmuebles.

3.4.3. Planificación temporal

Una vez visto de cerca las diferentes fases del proyecto nos detendremos en las principales tareas que las compondrán con el objetivo de realizar una evaluación conforme a su peso dentro de la planificación y acotarlas acorde a los tiempos estimados de su duración.

Descripción de las tareas

Fase 1: Fase inicial

Definición del alcance y contextualización

Entregable de GEP donde se define el alcance y la contextualización del proyecto.

Planificación temporal

Entregable de GEP donde se define la planificación temporal del proyecto.

Gestión económica y sostenibilidad

Entregable de GEP donde se define la gestión económica y la sostenibilidad del proyecto.

Condiciones de especialidad

Entregable de GEP donde se tratan y se definen los aspectos concretos de la especialidad de Software.

Presentación y documento final

Entregable final de GEP donde se recoge y presenta toda la especificación del trabajo dentro del alcance de GEP y se evalúa en conjunto a una presentación que englobe todos los aspectos del proyecto.

Aprendizaje y estudio de Xamarin

Aprendizaje del entorno Xamarin y adquisición de los conocimientos necesarios para el desarrollo de la APP.

Fase 2: Análisis y especificación de los requisitos

Análisis y especificación de los requisitos

Especificación de los requisitos de la aplicación móvil y entorno FAMA AFM.

Análisis y especificación de las funcionalidades

Especificación de las funcionalidades a implementar en la aplicación móvil y entorno FAMA AFM.

Evaluación

Evaluación de los requisitos y corrección modificación de los mismos.

Fase 3: Fase de desarrollo: Plan de Acción

Especificación de sistema y los servicios

Especificación del sistema y de los servicios a implementar.

Diseño de las pantallas y navegación

Diseño front-end de la aplicación y del programario FAMA AFM.

Especificaciones de seguridad

Especificación y revisión de los aspectos de seguridad.

Cobertura legal

Especificación de los aspectos legales de la aplicación.

Fase 3: Fase de desarrollo: Implementación

Aplicación

Implementación de la APP multiplataforma Android - IOS.

Comunicación servidor

Implementación de la comunicación APP-AFM FAMA.

Pruebas y evaluación

Pruebas de la APP y servicios de sistema servidor.

Fase 4: Fase final

Revisión final

Revisión exhaustiva del proyecto y de la documentación del trabajo.

Preparación de la defensa oral

Preparación de la presentación del proyecto ante tribunal.

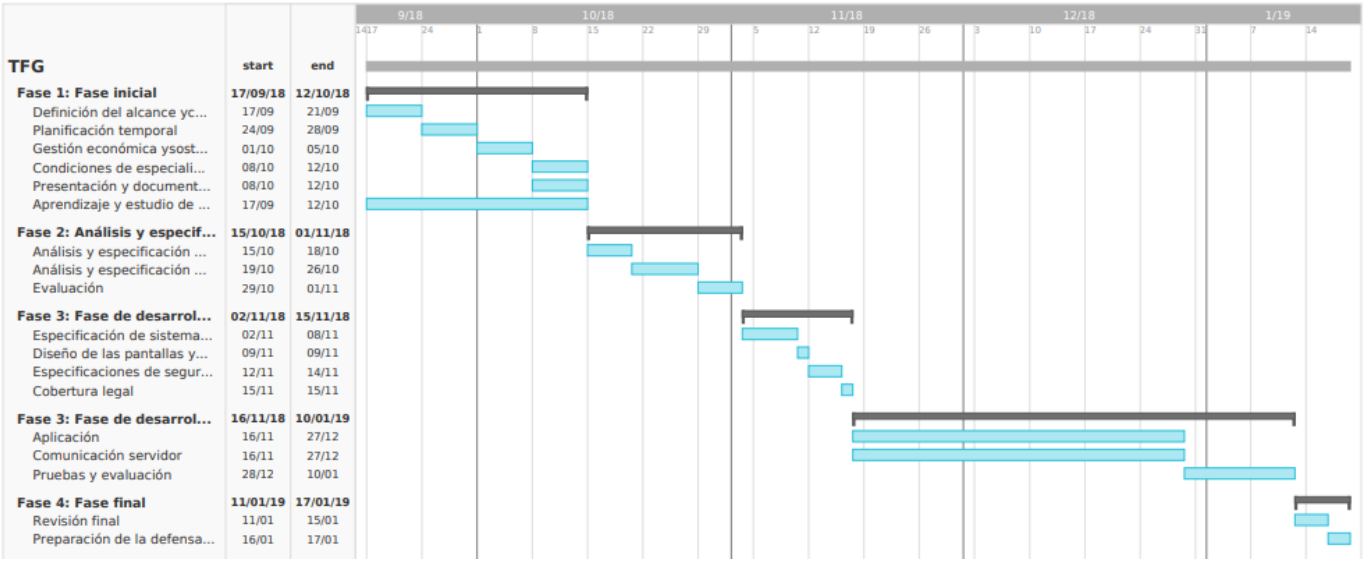
Tiempo estimado

En la tabla 1 muestra la estimación del número de horas dedicadas a cada tarea a realizar. Se estipula una dedicación de 5 horas al día para el desarrollo del proyecto.

Tarea	*Tiempo estimado (días)*	*Tiempo estimado (horas)*	*Peso Porcentual*
Definición del alcance y contextualización	5 días	24,00 h	5%
Planificación temporal	2 días	8,25 h	2%
Gestión económica y sostenibilidad	2 días	9,25 h	2%
Condiciones de especialidad	3 días	15,00 h	3%
Presentación y documento final	4 días	18,25 h	4%
Aprendizaje y estudio de Xamarin	5 días	25,00 h	5%
Total fase 1: Fase inicial	**20 días**	**99,75 h**	**20%**
Análisis y especificación de los requisitos	4 días	20,00 h	4%
Análisis y especificación de las funcionalidades	6 días	30,00 h	6%
Evaluación	4 días	20,00 h	4%
Total fase 2: Análisis y especificación de los requisitos	**14 días**	**70,00 h**	**14%**
Especificación de sistema y los servicios	5 días	25,00 h	5%
Diseño de las pantallas y navegación	1 días	5,00 h	1%
Especificaciones de seguridad	3 días	15,00 h	3%
Cobertura legal	1 días	5,00 h	1%
Total fase 3: Fase de desarrollo: Plan de Acción	**10 días**	**50,00 h**	**10%**
Aplicación	50 días	150,00 h	29%
Comunicación servidor	3 días	15,00 h	3%
Pruebas y evaluación	10 días	50,00 h	10%
Total fase 3: Fase de desarrollo: Implementación	**63 días**	**315,00 h**	**62%**
Revisión final	3 días	15,00 h	3%
Preparación de la defensa oral	2 días	10,00 h	2%
Total fase 4: Fase Final	**5 días**	**25,00 h**	**5%**
Total:	*102 días*	*509,75 h*	*100%*

Tabla 2: Tiempo estimado de cada tarea

Diagrama de GANTT



La figura 1 muestra la planificación de las tareas en un diagrama de Gantt.

Figura 4: Diagrama de Gantt del proyecto. Generado con https://prod.teamgantt.com/

3.4.4. Alternativas y plan de acción

Como se ha señalado en apartados anteriores, el desarrollo del proyecto está sujeto a cambios que pueden alterar la planificación.

Debido a los ciclos de trabajo de la fase 2 y la fase 3 de la planificación, podremos revisar y adaptar dinámicamente la duración de las diferentes tareas, alargando el tiempo de desarrollo o, rara vez, acortando y dando paso a la siguiente en los casos pertinentes. Por otro lado, si una tarea dura más de lo esperado, una tarea posterior tendrá que acortarse o, en el peor de los casos, replantear parte del proyecto para o bien omitirla o bien mutarla en otra tarea de menor tiempo de desarrollo.

Haciendo una primera valoración de los tiempos de desarrollo estipulados, la fase del proyecto que puede generar más retrasos es la fase de desarrollo, la cual aglomera el mayor volumen de horas del proyecto. De esta manera se espera cubrir todos los imprevistos que puedan surgir.

Cabe añadir que, en la planificación no se han añadido fines de semana, dejando ese margen para horas adicionales que se deban dedicar a ciertos aspectos del proyecto de forma puntual. Por otra parte, entre la finalización del proyecto y la entrega final también se ha dejado otro margen de tiempo por si fuese necesario alargar alguna de las fases más allá de las horas planificadas.

A continuación, se mencionan algunos ejemplos de posibles fuentes de posibles riesgos.

Riesgo	*Gravedad*	*Plan de contingencia*
Complejidad del entorno FAMA AFM	Baja	Posibilidad de consulta a trabajadores experimentados en la plataforma y ayuda en el desarrollo de las funcionalidades relacionadas.
Especificaciones inviables o de complejidad extrema	Alta	Realización de reuniones de planificación semanales que permitan desviar el rumbo de ciertas tareas o la eliminación de las mismas de una manera rápida evitando una mayor pérdida de tiempo.
Falta de tiempo de desarrollo	Media	Realización de reuniones de evaluación semanales para la modificación de las especificaciones y/o el restablecimiento de los tiempos de desarrollo.
Bugs	Media	Tiempo de desarrollo amplio con tiempo suficiente para dedicar en los distintos errores que puedan surgir en el proyecto.
Pérdida del trabajo realizado	Alta	Copias en el SVN de la empresa para realizar copias de seguridad de forma habitual para la recuperación de los datos en caso de pérdida.
Desconocimiento del framework de desarrollo	Alta	Realización de cursos online y ayuda a través de foros de internet, comunidad activa y del soporte oficial de Microsoft.

Tabla 3: Potenciales riesgos del proyecto, ponderación y plan de contingencia para los mismos.

3.5. Presupuesto

Una vez hemos definido el contexto y alcance del proyecto y una planificación para las diferentes partes que componen el reto, ya nos vemos en posición de poder identificar los diferentes costes en los que se devendrá y cuantificarlos para poder estudiarlo de una forma más concreta de cara al futuro. El objetivo será presupuestar estos costes y clasificarlos según sean costes directos, indirectos o inesperados para poder prever estos en los diferentes ámbitos del proyecto.

3.5.1. Costes directos

Los costes directos son los costes que guardan relación estrecha con el proyecto. Son costes que se establecen en las primeras fases de concepción y suelen verse reflejados en los presupuestos o estimaciones de costes.

En el cálculo de la amortización, tendremos en cuenta la vida útil del producto y el tiempo de duración del proyecto, que será el tiempo donde serán usados estos productos.[13] [14] En concreto usaremos la siguiente fórmula para establecer dicho valor.

$$Cuota\ de\ amortización = \frac{Valor\ de\ adquisición - Valor\ residual}{Años\ de\ vida\ útil}$$

Una vez calculado la cuota de amortización, calcularemos la amortización en cuanto al proyecto dividiendo por la durada del proyecto en la que vayan a ser usados correspondientemente.

Presupuesto de hardware

La tabla 1 contiene los costes de hardware que van a ser usados en el desarrollo del proyecto.

Producto	*Precio (€)*	*Unidades*	*Vida útil (años)*	*Amortización*
iMac Pro	2.255,59 €	1	4 años[15]	281,95 €
IPhone 7	374 €	1	4 años[15]	46,75 €
LG Nexus 4	287,7 €	1	4 años	35,96 €
Total	*2917,29 €*			*364,66 €*

Tabla 4: Presupuesto de hardware detallado.

Cabe destacar que, todos los dispositivos usados durante el desarrollo ya han sido utilizados en anteriores proyectos y no han sido comprados de forma específica para este. En concreto, nos encontramos en el segundo año de amortización de estos gastos.

[13] "Calcular amortización de un producto" https://tutobasico.com/calcular-amortizacion/
[14] "Activos, amortización y depreciación" https://debitoor.es/guia-pequenas-empresas/contabilidad/activos-amortizacion-y-depreciacion
[15]" La vida útil real: cómo determinar cuando la renovación de un dispositivo de Apple pasa de ser capricho a necesidad" https://www.applesfera.com/iphone/la-vida-util-real-como-determinar-cuando-la-renovacion-de-un-dispositivo-de-apple-pasa-de-ser-capricho-a-necesidad

Presupuesto de software

La tabla 2 muestra el resumen de los costes del software necesario para la realización del proyecto.

Producto	*Precio (€)*	*Unidades*	*Vida útil (años)*	*Amortización*
Visual Studio Community	0 €	-	-	-
Visual Studio Community for Mac	0 €	-	-	-
Microsoft Office 2016 Pro Plus	175 €	1	5 años[16]	17,5 €
PLSQL Developer 50 User Licence	2022 € (40,44 €/u)	50 Licencias (Usadas 1)	6 años[17]	3,37 €
Eclipse Neon 2.0	0 €	-	-	-
Microsoft Visual Sourcesafe	0 €	-	-	-
Windows 10 Professional License	259 €	Licencia múltiple	5 años	21,58 €
Total	*474,44 €*			*42,45 €*

Tabla 5: Presupuesto de software detallado.

Aunque las licencias usadas son de por vida, la vida útil se ha calculado según el soporte y la periodicidad de nuevas versiones que contengan utilidades necesarias y vitales según los criterios del departamento de sistemas de FAMA Systems.

Cabe destacar, que al igual que con el presupuesto de hardware, las licencias de los productos usados en el desarrollo ya han sido adquiridas con anterioridad.

Presupuesto de recursos humanos

La tabla 3 muestra el presupuesto de los recursos humanos de los diferentes profesionales involucrados en el proyecto.

[16] "Versiones de Office y conjuntos de requisitos – Microsoft" https://docs.microsoft.com/es-es/office/dev/add-ins/develop/office-versions-and-requirement-sets
[17] "Licenciamiento de PL/SQL Developer – ComponentSource" https://www.componentsource.com/es/product/pl-sql-developer/licensing

Producto	*Precio (€/horas)*	*Tiempo (horas)*	*Personas*	*Coste total*
Director de proyecto	20,83 €/h[18] [19]	100 h	1 Director	2083 €
Diseñador y analista	18,22 €/h[6] [7]	100 h	1 Diseñador - Analista	1822 €
Programador	14,06 €/h[6] [7] [20]	365 h	1 Programador	5131,9 €
Administrador de sistemas	18,22 €/h[6] [7]	50 h	1 Técnico de sistemas	911 €
Total			*4*	*9947,9 €*

Tabla 6: Presupuesto de recursos humanos detallado.

El precio por hora de los técnicos involucrados en el proyecto se ha calculado mediante una aproximación dentro de las tablas salariales establecidas en otros proyectos y diferentes informes del sector.

3.5.2. Costes indirectos

Los costes indirectos son los que se relacionan de manera tangencial con el proyecto y las tareas previstas. Son costes que no afectan directamente a los diferentes aspectos del plan pero si a los ítem que se usan dentro de estos.

La tabla 4 muestra los costes indirectos derivados de los diferentes aspectos del proyecto.

Producto	*Precio*	*Unidades*	*Coste estimado (€)*
Licencia desarrollador Android/Google Play	22 €	1	22,00 €
Licencia desarrollador iOS	99 €/año	1 año	99,00 €
Consumo energético	0,17 €/kWh[21]	450 h aprox.	76,50 €
Internet	20 €/mes	5 meses	100,00 €
Azure Notification Hubs	101,28 €	1	12,66 €[22]
Curso de programación Xamarin	-	1	-

[18] "BOE" https://www.boe.es/boe/dias/2009/04/04/pdfs/BOE-A-2009-5688.pdf
[19] "Pisos salariales mínimos – Unión informática" https://unioninformatica.org/institucional/convenio-colectivo-de-trabajo/
[20] "La realidad del perfil de informático júnior en España según los informes – Xataka" https://www.xataka.com/tecnologiazen/la-realidad-del-perfil-de-informatico-junior-en-espana-segun-los-informes
[21] "Precio neto de la electricitdad – Mincotur" https://www.mincotur.gob.es/es-es/IndicadoresyEstadisticas/BoletinEstadistico/IV.%20Energ%C3%ADa%20y%20emisiones/IV_12.pdf
[22] Coste estimado con una amortización a 5 y dentro la durada del proyecto.

Espacio de trabajo	1500 €/mes	110 m^2	190,91 €[23] [24]
Total			*501,07 €*

Tabla 7: Presupuesto de los costes indirectos detallado.

En un principio el curso de Xamarin iba a suponer un coste dentro del proyecto pero después de una valoración de posibles alternativas dentro del proceso de concepción y contextualización, se determinó que la calidad de documentación gratuita era suficiente para el desarrollo del proyecto.

3.5.3. Costes inesperados

Los costes inesperados los conformarán todos aquellos posibles costes que no se hayan previsto i vengan derivados de los distintos factores de riesgo que hayamos identificado o de directamente, imprevistos que hayan surgido durante el desarrollo.

En un principio, los costes inesperados del proyecto podrían surgir principalmente a raíz de licencias dentro de librerías o utilidades necesarias que vayan siendo requeridas en el desarrollo de la aplicación, aunque el alcance tecnológico ya ha sido medido y estudiado y es poco probable que surjan, relegando este coste a quedar cubierto por los costes de contingencia.

3.5.4. Costes de contingencia

Antes de reunir todo los costes estipulados hasta ahora, determinaremos un coste de contingencia para cubrir gastos inesperados que puedan surgir durante la realización del proyecto.

Este coste lo estipularemos como un porcentaje del total con el objetivo de alcanzar todas las distintas áreas lo fijaremos en un 8%[25] [26] [27] teniendo en cuenta que la mayoría de los costos fijos ya han sido costeados.

3.5.5. Coste total y gestión de control

Habiendo desgranado todos los costes en los anteriores apartados, en la tabla 5 se muestran todos los costes identificados y estipulados en el proyecto de forma resumida.

[23] Coste estimado con un espacio personal de trabajador estándar de 2,3 m^2 dentro la durada del proyecto.
[24] "La oficina ideal"
https://cincodias.elpais.com/cincodias/2014/10/28/pyme/1414500383_553511.html
[25] "Determinar el impacto de los riesgos en el costo del proyecto mediante el VME"
https://guiadeproyecto.com/2015/01/12/determinar-el-impacto-de-los-riesgos-en-el-costo-del-proyecto-mediante-el-vme/
[26] 12 técnicas para la estimación de costes en proyectos – Tendencias & Innovación, Marc Bara"
https://www.obs-edu.com/es/blog-investigacion/project-management/12-tecnicas-para-la-estimacion-de-costes-en-proyectos
[27] "Reservas de contingencia y reservas de gestión – Jose Huerta"
https://josehuerta.es/gestion/proyectos/costes/reservas-de-contingencia-y-reservas-de-gestion

Concepto	*Coste (€)*
Presupuesto de hardware	364,66 €
Presupuesto de software	42,45 €
Presupuesto de recursos humanos	9947,9 €
Total Costes directos	**10.355,01 €**
Total Costes indirectos	**501,07 €**
Subtotal	**10.856,08€**
Costes de contingencia (8%)	868,49 €
Total	*11.724,57 €*

Tabla 8: Costes totales detallados.

Como hemos destacado en el apartado de costes de contingencia, la mayoría de costos del proyecto los cubrimos con el gasto de la empresa FAMA Systems, la cual ya cuenta con las licencias, hardware y personal de proyectos para cubrir los diferentes aspectos del trabajo.

Por otra parte, dentro de la misma, el desarrollo de aplicaciones es muy joven, y de ahí la mención de los costes inesperados derivados del aspecto técnico de la aplicación que, aunque se ha realizado un estudio previo, hay que añadir el factor de desconocimiento en el entorno.

3.6. Sostenibilidad y Compromiso social

En esta sección evaluaremos la sostenibilidad del proyecto en las diferentes áreas de impacto: El ámbito económico, el área ambiental y el entorno social.

3.6.1. Dimensión económica

Dentro del ámbito económico del proyecto, se ha realizado un estudio en profundidad de las necesidades del desarrollo y de las diferentes herramientas siempre se han buscado las más duraderas, fiables y estrictamente necesarias.

Huelga decir que, en el caso de los costes directos y de hardware, estos ya estaban cubiertos por anteriores proyectos dentro del desarrollo de aplicaciones aunque todavía nos encontramos en el periodo de amortización de dichos costes.

En cuanto al coste total estimado del proyecto, en comparación a otros proyectos de ámbito tecnológico de desarrollo de aplicaciones, nos encontramos con un presupuesto muy ajustado, acorde a las especificaciones y el desarrollo en con para la empresa más cerca de un coste de I+D más que a un proyecto más dentro del alcance de la empresa.

Si hablamos de su concepción, este proyecto ya se ideó con la máxima de reducir costes, evitando el doble desarrollo multiplataforma y usando un framework integrador que permitiese trabajar paralelamente en los diferentes entornos en los que se quiere desplegar la APP.

De la misma forma, si hablamos de recursos humanos, contamos con los mínimos integrantes requeridos para garantizar el desarrollo y la calidad del mismo dentro de la empresa, con actores llevando a cabo diferentes roles dentro del proyecto.

3.6.2. Dimensión ambiental

Durante el desarrollo del proyecto, se usarán distintos computadores para poder cubrir las diferentes áreas, tanto de APP como servidor, requeridas para el desarrollo.

Estos costes realmente son remarcables, pues conforman dispositivos físicos que afectan directamente al aspecto medioambiental. En este aspecto cabría enfatizar la existencia de posibles alternativas al despliegue de medios del proyecto, como el alquiler de máquinas para el desarrollo IOS por ejemplo[28] [29], pero en este aspecto se estudiaron y valoraron los beneficios de una inversión a largo plazo, lo que se traduce en un ahorro para actuales y futuros proyectos por el hecho de tener un hardware confiable y duradero y no comprar el justo para el desarrollo del proyecto con una futura y casi segura inversión si hubiese sido el caso.

Por otro lado, se están reutilizando recursos propios de la empresa, como los dispositivos móviles usados en antiguos proyectos y los propios computadores con Windows, ya usados anteriormente para proyectos de desarrollo en el ámbito de servidor.

En cuanto al servidor, destacaremos que ya usaremos un entorno activo en la actualidad y no será necesario hacer un despliegue adicional para este proyecto, no generando ningún gasto ni consumo de recursos adicionales.

Como resumen, remarcar lo expuesto en este apartado; existe un margen para la reducción de costes y recursos usados en el proyecto pero, si pensamos en futuros proyectos y la idea de que convivan distintos proyectos de desarrollo de aplicaciones al mismo tiempo, el despliegue está justificado y sigue siendo ajustado.

3.6.3. Dimensión social

Dentro del entorno social, distinguiremos tres aspectos diferenciados relacionados con tres de los actores principales del proyecto: la empresa, mi experiencia personal y el usuario final.

Para la empresa, este proyecto ha significado un paso más en la expansión dentro del mundo del desarrollo de aplicaciones. Este trabajo conforma el tercero de

[28] "Xcode para Windows con Smarface para desarrollo nativo de ios – Smartface" https://www.smartface.io/xcode-para-windows-con-smartface-para-desarrollo-nativo-de-ios/
[29] "Mac rental service that provides a PC an Mopbile users remote Access ro our Mac servers through the Cloud – Macincloud" https://www.macincloud.com/

la empresa donde se desarrolla una APP y el segundo que sigue la metodología descrita en el contexto y que hace uso de la inversión que se realizó en las herramientas mencionadas anteriormente en el presupuesto.

Este proyecto conforma también gran parte del I+D de la empresa en la que se han tenido que definir diferentes metodologías tanto de configuración de entornos y uso de recursos como de desarrollo por la inexistencia de estos anteriormente.

Ligado con mi experiencia personal, el trabajo está resultando exigente, ya que las decisiones que se toman han de ser consensuadas y seguras, a la vez que los pasos a seguir y el entorno de desarrollo se están definiendo desde cero y han de ser reproducibles y no vale con tan solo configurar los distintos componentes i programar la aplicación. Este proyecto tiene mucho trabajo previo más ligado a un profesional de sistemas que el propio a un desarrollador de software con decisiones importantes y con influencia directa en futuros proyectos.

Por otra parte, he de afirmar que dicha experiencia hace que me mueva en muchos ámbitos diferentes que, aparte de aprender lo propio en el desarrollo de la APP, hace que coja experiencia y una visión amplia de todo el contexto del desarrollo tanto del proyecto como de, en parte y en lo que me concierne, de la propia empresa.

Por último, si hablamos del propio cliente y de los usuarios a los que va destinado este proyecto, hace falta hablar de la herramienta FAMA AFM de la empresa. Este software, como hemos detallado en anteriores apartados, es un software de *Facility Management*, que procura facilitar la gestión de diferentes aspectos de una empresa, organismo u organización.

Siguiendo esta filosofía, los usuarios de la aplicación son usuarios de diferente índole y distintos entornos y contextos, lo que nos hace pensar en abarcar las diferentes situaciones que estos puedan llegar a experimentar. Más en concreto, debido a la naturaleza del software FAMA AFM, estos se ven siempre obligados a la gestión de los diferentes recursos a través de la web, siendo este un proceso algo lento en el contexto en que nos movemos, donde como se presentó en el contexto, gran parte de la población, y por ende, trabajadores, cuentan con un dispositivo móvil a mano.

Esto nos lleva a afirmar que, el desarrollo de la app es, además de una evolución lógica del programa de la empresa, una herramienta muy poderosa y útil para el usuario.

3.6.4. Matriz de sostenibilidad

Con el objetivo de evaluar y garantizar el desarrollo de un sistema sostenible en todos lo sejes, se ha evaluado el impacto del proyecto de forma sistemática en el entorno que se enmarca desde las perspectivas ambientales, económicas y sociales. El análisis resume lo expuesto en los apartados anteriores dentro de la tabla 6 y

responde a unas preguntas mediante las cual se evalúan los diferentes aspectos del proyecto[30].

	PPP	*Vida útil*	*Riesgos*
Económico	Consumo de diseño	Huella ecológica	Ambientales
	8/10	8/10	2/10
Ambiental	Factura	Plan de viabilidad	Económicos
	6/10	8/10	4/10
Social	Impacto personal	Impacto social	Sociales
	9/10	7/10	0/10
Rango de sostenibilidad	23/30	23/30	1-(6/30)
		70/90	

Tabla 9: Matriz de sostenibilidad del proyecto.

[30] "Mòdul 2.6 - El informe de sostenibilidad 2018 – Facultat d'Informàtica de Barcelona" Páginas 2-4

4. Análisis funcional y especificación de los requisitos

Aunque la metodología propuesta está más cerca de una metodología ágil, con diferentes iteraciones y con la mínima documentación y especificación, siempre es necesario fijar unos requisitos invariables que han de formar la base del proyecto y han de estar presentes como columna vertebral del mismo.

En esta sección desgranaremos dichos objetivos, los describiremos y veremos los cambios realizados en el software ya existente FAMA/AFM así como el esquema general de la APP.

4.1. Requisitos de la App

Como hemos comentado anteriormente, a continuación se muestran los grandes objetivos transversales del desarrollo:

Dichos objetivos marcarán en grandes rasgos como ha de ser la aplicación y cuál es el uso esperada de ella.

Los objetivos principales de la aplicación son:

- Sistema multiplataforma: iOS + Android.
- Multiidioma: tiene que mostrar las etiquetas en el idioma que esté configurado el dispositivo. Tiene que estar traducido al español, catalán e inglés. En caso que el dispositivo esté configurado en un idioma diferente se mostrará en inglés.
- Descargable des de Apple Store y Google Play.
- Instalación gratuita.
- Comunicación mediante servicios REST.
- Realizar Reservas de salas.
- Consultar Reservas de salas.
- Modificar o anular reservas pendientes.
- Seleccionar Edificio (de las salas) mediante GPS del dispositivo.
- Permitir seleccionar sala filtrando por el equipamiento disponible: Proyector, Audioconferencia, Videoconferencia, WiFi y PC.
- Inicio de sesión contra varios servidores mediante distintas URLs.
- Sistema usable, con curva de aprendizaje corta e intuitiva.

4.2. Especificación funcional

Una vez definidos unos requisitos generales, identificaremos los requisitos funcionales.

4.2.1. Casos de uso

UC-01. Iniciar sesión en un servidor

La APP debe poder iniciar sesión contra diferentes servidores. Se deberá poder configurar la URL a la que se debe conectar.

UC-02. Cerrar sesión

La APP debe permitir al usuario cerrar la sesión y regresar a la pantalla de inicio de sesión.

UC-03. Realizar una reserva de sala

El usuario debe poder realizar una reserva de sala y registrarla en el sistema desde la APP.

UC-04. Consultar una reserva de sala propia

El usuario debe poder consultar una reserva de sala que haya registrado en el sistema.

UC-05. Consultar reservas de sala futuras propias

El usuario debe poder consultar todas las reservas de sala que haya registrado en el sistema que no hayan finalizado en la fecha actual.

UC-06. Consultar histórico de reservas de sala propias

El usuario debe poder consultar todas las reservas de sala que haya registrado en el sistema que hayan finalizado en la fecha actual.

UC-07. Consultar reservas de una sala en un día

El usuario debe poder ver la ocupación de una sala en un día concreto para reservar en una franja horaria disponible.

UC-08. Consultar reservas de sala propias del día

El usuario debe poder ver todas las reservas que haya registrado en el sistema en la fecha actual de una manera rápida y accesible.

UC-09. Consultar salas disponibles para reservar cercanas

El usuario debe poder ver una lista de salas disponibles y cercanas en la fecha actual de manera que pueda acceder a su reserva de una manera rápida y accesible.

UC-10. Búsqueda de salas disponibles por criterios

El usuario debe poder realizar una búsqueda de salas disponibles mediante unos criterios en una fecha concreta.

Los criterios de búsqueda serán:

- Día de la reserva

- Duración de la reserva
- Fecha inicio y final de margen para la reserva
- Capacidad de la sala
- Servicios de la sala

Los servicios de las salas en el sistema serán: Proyector, Audioconferencia, Videoconferencia, WiFi y PC.

UC-11. Modificar una reserva de sala propia

El usuario debe poder modificar una reserva de sala que haya registrado en el sistema y que no haya finalizado.

UC-12. Dar de baja una reserva de sala propia

El usuario debe poder dar de baja una reserva de sala que haya registrado en el sistema y que no haya finalizado ni esté actualmente en transcurso.

UC-13. Finalizar una reserva de sala propia en transcurso

El usuario debe poder finalizar una reserva de sala que haya registrado en el sistema y que esté actualmente en transcurso.

UC-14. Consultar información "Acerca de" de la aplicación

El usuario debe poder ver la información del sistema con ciertos aspectos de la configuración así como el enlace a la web de la empresa y a su política de privacidad.

UC-15. Consulta de Inmuebles del sistema

El usuario debe poder consultar una lista de los inmuebles del sistema con salas disponibles para reservar.

UC-16. Acceder a la web de contacto de la empresa

El usuario debe poder acceder a la página de contacto de la empresa desde la aplicación de una manera rápida y accesible.

UC-17. Acceder a la web de Política de privacidad de la empresa

El usuario debe poder acceder a la página de política de privacidad de la empresa de una manera rápida y accesible.

4.2.2. Diagrama de casos de uso

A continuación se muestra el diagrama de casos de uso con los diferentes actores del sistema.

En él aparecen ciertos casos de uso que se escapan del alcance de este proyecto pero se muestran con el objetivo de poder apreciar la magnitud y el comportamiento del sistema de forma global.

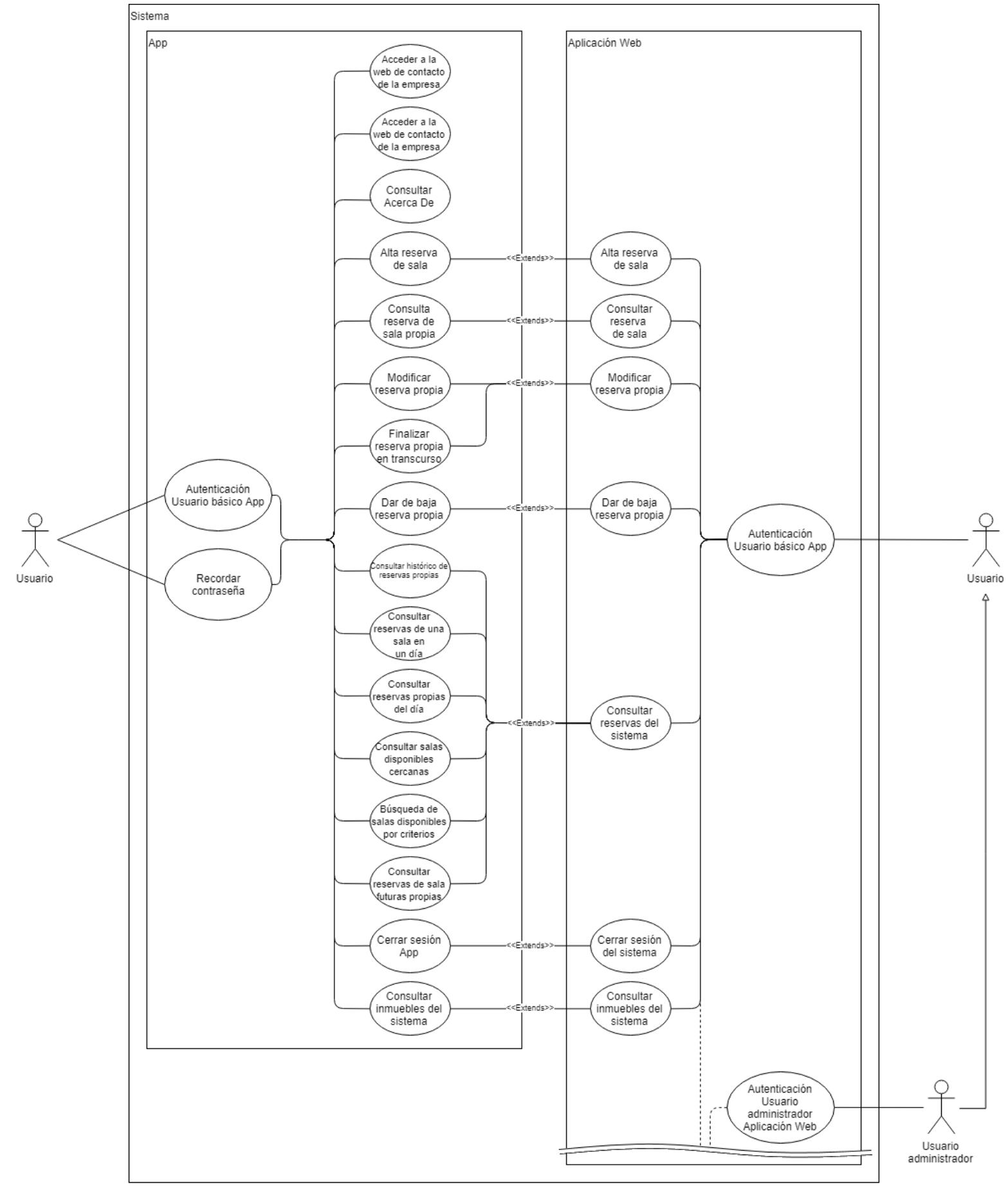


Ilustración 1: Fragmento del diagrama de casos de uso de la aplicación y la aplicación web.

4.3. Especificación no funcional

Una vez definidos los requisitos funcionales, identificaremos los requisitos no funcionales. Dentro de los requisitos no funcionales diferenciaremos los requisitos técnicos, más concretos y de carácter más específico.

4.3.1. Requisitos no funcionales

NFR-01. Apariencia simple

La aplicación debe tener un acabado simple y cercano a los usuarios.

NFR-02. Eficiencia

La aplicación debe mostrar la información de manera rápida y accesible. La información mostrada será útil y simple.

NFR-03. Multiidioma

La aplicación ha de estar traducida al español, catalán e inglés. Por defecto el idioma a mostrar será el inglés.

NFR-04. Descarga de la aplicación desde la Apple Store

La aplicación ha de poder descargarse desde la tienda de aplicaciones de Apple.

NFR-05. Descarga de la aplicación desde la Google Play

La aplicación ha de poder descargarse desde la tienda de aplicaciones de Google.

NFR-06. Instalación gratuita

La aplicación será de descarga e instalación gratuitas.

NFR-07. Usable

La aplicación debe tener un diseño eficiente y sencillo.

NFR-08. Aprendizaje intuitivo

La aplicación debe tener una curva de aprendizaje corta e intuitiva.

NFR-09. Privacidad

Ningún usuario podrá acceder a los datos personales de los otros. No se podrá conocer el usuario que realizó otra reserva de sala que no sea propia.

NFR-10. Online

La aplicación solo será operativa de forma online. Sin conexión la aplicación no mostrará datos y no permitirá modificarlos.

NFR-11. Creación de sesión

Al iniciar sesión, la aplicación se conectará al servidor y registrará una entrada en la tabla de sesiones. Las siguientes comunicaciones usarán el identificador de la sesión registrada.

NFR-12. Consumo de licencia

Al iniciar sesión, la aplicación se conectará al servidor y asignará una licencia a la sesión del usuario.

NFR-13. Liberación de licencia

Un administrador de forma manual podrá liberar una licencia bloqueada por un usuario.

NFR-14. Inmueble por defecto

La aplicación cargará la información principal con el inmueble vinculado al usuario en el sistema.

4.3.2. Requisitos técnicos

NFR-15. Aplicación Android

La aplicación estará disponible en sistemas Android.

NFR-16. Aplicación iOS

La aplicación estará disponible en sistemas iOS.

NFR-17. Comunicación REST Servidor

La aplicación se comunicará con el servidor mediante arquitectura REST.

NFR-18. Comunicación mediante JSON

La comunicación con el servidor será mediante el uso de objetos JSON.

NFR-19. Uso de GPS

La aplicación deberá hacer uso del GPS para mostrar los espacios más cercanos al usuario.

NFR-19. Patrón Modelo-Vista-Modelo de Vista

El proyecto estará estructurado con el paradigma Modelo-Vista-Modelo de Vista con las características propias del entorno Xamarin.

NFR-20. Guardar datos temporales

La aplicación recordará la sesión del usuario e iniciará sesión automáticamente.

De la misma manera, guardará también algunos datos muy usados por la aplicación.

NFR-21. Borrar datos temporales

Cuando el usuario inicie sesión, la aplicación detectará si el usuario o el servidor han variado de la última vez que se inició sesión y eliminará los datos temporales existentes si fuese el caso.

4.4. Cambios en el FAMA/AFM

En esta sección desgranaremos los cambios necesarios en la aplicación web y el servidor para el funcionamiento de la reserva de salas.

Cierta parte de la funcionalidad ya se encontraba parcialmente implementada en la aplicación web aunque ha hecho falta una puesta a punto y ciertas actualizaciones para equiparar todas las funcionalidades de ambas aplicaciones.

Así mismo, algunos de los siguientes apartados no podrán ser muy extensos al tratar detalles internos del programario de la empresa aunque se tratará de hacer la máxima aproximación para poder entender la naturaleza de dichos cambios y de la implementación.

4.4.1. Cambios en el modelo de datos

Sala

Dentro de la estructura de la *sala* en servidor, se crearán nuevos campos para especificar los diferentes *servicios* fijos asociados a la sala. Se marcará por cada *servicio* fijo si la sala dispone de él o no.

Servicios de sala

Se creará la estructura necesaria para persistir los diferentes servicios disponibles de las salas y se crearán cinco servicios básicos: *Proyector, Video Conferencia, Audio Conferencia, Wifi y PC.*

En concreto, se incluirán 5 campos correspondientes a los 5 tipos de servicios fijos que podrán asociarse a una sala. Por defecto tendrán el valor 'N'.

Reserva de sala

Se añadirá un campo para diferenciar si la reserva de la sala ha sido creada desde la aplicación móvil.

En concreto, se incluirá el campo "Alta desde APP". Por defecto tendrá el valor 'N'.

4.4.2. Casos de uso de la aplicación Web

Nueva entrada en el Menú de APP Salas

El usuario debe poder acceder a la pantalla de configuración de parámetros y al listado de sesiones activas de la aplicación de Reserva de Salas desde el menú.

Seguridad por licencia y roles

Un usuario sin permisos no podrá acceder a la entrada de la aplicación de salas. Se ha de crear una licencia de usuario para este módulo y se han de crear roles para la visibilidad y accesibilidad a dicho módulo en la aplicación web.

Identificación del origen de una reserva de sala

El usuario debe poder ver si una reserva de una sala se ha creado desde la aplicación o no.

En concreto se realizarán los siguientes cambios:

- En la ficha de una reserva:
  - Se incluirá el icono de un móvil en caso de 'S'.
- En el listado de reservas:
  - Se incluirá una columna "APP" que pondrá el icono de un móvil en el caso de tratarse de una alta desde APP

Filtrado de reservas de ralas por su origen

El usuario debe poder filtrar en una lista de reservas de sala por el sistema en el que se realizó la reserva mediante un filtro.

En el listado de reservas se incluirá el filtro “Altas desde APP” con los valores ‘S’, ’N’, ’Todas’.

Identificación servicios fijos de una sala

El usuario debe poder identificar de una manera directa y sencilla de qué servicios fijos dispone una sala.

En concreto se realizarán los siguientes cambios:

- En la ficha de una sala:
  - Se incluirán 5 campos tipo *checkbox* correspondientes a los 5 tipos de servicios fijos con un icono representativo, bloqueados y marcados en el caso de que la sala disponga de dicho servicio.
- En el listado de salas:
  - Se incluirá una columna “Servicios fijos” donde aparecerán los iconos representativos de los servicios de los cuales disponga la sala.

Asignación servicios fijos de una sala

El usuario debe poder indicar de una manera directa y sencilla de qué servicios fijos dispone una sala.

Dichos servicios estarán identificados con un icono representativo y un marco estilo *checkbox* que el usuario podrá marcar en el caso que la sala disponga del servicio.

4.4.3. Servicios web

A continuación listaremos los servicios web de la aplicación. Distinguiremos los casos de uso en tres tipos: los servicios web genéricos, los servicios web de reserva de salas y los servicios web de espacios. Realizamos dicha distinción ya que dentro de la aplicación web, conceptualmente y tal y como está estructurado internamente, están diferenciados.

Servicios web genéricos

WSC-01. Login genérico

La aplicación validará el usuario y la contraseña en el servidor indicado e iniciará sesión en el servidor.

El servidor devolverá la información de sesión y ciertos datos del usuario.

WSC-02. Validar sesión

La aplicación enviará la información de la sesión obtenida en el inicio de sesión o que tenga guardado en la caché y el servidor comprobará que sea actual y correcta.

En el caso de no coincidir con el servidor, la aplicación informará al usuario.

WSC-03. Test de conectividad

La aplicación realizará un ping a una página perenne en la web para comprobar la disponibilidad de la red.

En cualquier caso se informará al usuario para que compruebe la red o el servidor de conexión.

WSC-04. Consultar valor constante

La aplicación podrá realizar una consulta al servidor con el nombre identificador de una constante.

La aplicación web devolverá el valor de la constante en el servidor con el identificador recibido.

WSC-05. Descargar documento

La aplicación podrá hacer la petición por un documento con el identificador de un documento.

La aplicación web devolverá el documento con el identificador recibido o un error 404 en el caso de que no exista dicho documento en el servidor.

Servicios web de reserva de salas

WSC-06. Consultar salas disponibles

La aplicación podrá consultar al servidor la lista de salas disponibles.

La aplicación web consultará en el servidor la lista de salas disponibles en un tiempo cercano a la consulta y devolverá las primeras diez salas de la consulta.

WSC-07. Consulta reservas de sala del usuario

La aplicación podrá consultar al servidor las reservas del usuario filtradas con algunos criterios.

La aplicación web devolverá las reservas del usuario que cumplan los criterios especificados. Los criterios serán: reservas anteriores a la fecha actual, reservas posteriores a la fecha actual y reservas en el día de hoy.

WSC-08. Alta reserva de sala

La aplicación podrá realizar un intento de alta de una reserva de sala en el sistema.

La aplicación web realizará el intento de alta de la reserva y devolverá un “ok” en el caso que se haya realizado o un error con un mensaje que la aplicación mostrará al usuario en el caso contrario.

WSC-09. Modificación reserva de sala del usuario

La aplicación podrá modificar una reserva de sala ya existente en el sistema.

La aplicación web realizará la modificación y devolverá un “ok” en el caso que se haya realizado correctamente o un error con un mensaje que la aplicación mostrará al usuario en el caso contrario.

WSC-10. Anular reserva de sala del usuario

La aplicación podrá dar de baja una reserva de sala.

La aplicación web dará de baja la reserva y devolverá un "ok" en el caso que se haya podido eliminar o un error con un mensaje que la aplicación mostrará al usuario en el caso contrario.

Servicios web de espacios

WSC-22. Consultar espacios

La aplicación podrá consultar al servidor la lista de espacios con salas reservables.

La aplicación web consultará en el servidor la lista de espacios con salas ordenadas por cercanía al usuario y las devolverá a la aplicación.

4.5. Esquema de la APP

Finalmente analizaremos la estructura y los paradigmas utilizados para ordenar y organizar el código de la aplicación.

Dentro de la libertad que he tenido para realizar este proyecto, estos apartados han sido de los más notables en cuanto al impacto dentro del proyecto y la importancia de cara a futuros proyectos de la misma índole.

Es necesario definir patrones que permitan la modularidad y la flexibilidad para diversas aplicaciones y que sean fácilmente reproducibles así como una estructura que englobe las diferentes secciones y capas que vaya a tener la aplicación.

4.5.1. Patrón modelo-vista-modelo de vista

Al empezar a trabajar con C# en Xamarin, mucha documentación y muchos sitios especializados tratan con el patrón Modelo-Vista-Modelo de vista, sin embargo, aunque fue el patrón de programación elegido a nivel de organización de código, se estuvo sopesando otros patrones usados y extendidos en este entorno.

Alternativas y comparativa

El principal competidor es el patrón **Modelo-Vista-Controlador** (de ahora en adelante MVC). En este patrón de diseño se separan los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. Cuando la lógica de negocio realiza un cambio, es necesario que ella sea la que actualiza la vista[31].

Por otra parte, tenemos el patrón escogido, el patrón **Modelo-Vista-VistaModelo** o Modelo de vista (de ahora en adelante MVVM). En este patrón de diseño se separan los datos de la aplicación de la interfaz de usuario pero, en vez de controlar manualmente los cambios en la vista o en los datos, estos se actualizan directamente cuando sucede un cambio en ellos. Por ejemplo, si la vista actualiza un dato que está presentando se actualiza el modelo automáticamente y viceversa[31].

[31] "¿Qué son MVC y MVVM? – Adictos al trabajo" https://www.adictosaltrabajo.com/2012/10/07/zk-mvc-mvvm/

Las principales diferencias entre MVC y MVVM son que en MVVM el *Controller* cambia a *ViewModel* y hay un "binder" que sincroniza la información en vez de hacerlo un controlador *Controller* como sucede en MVC.

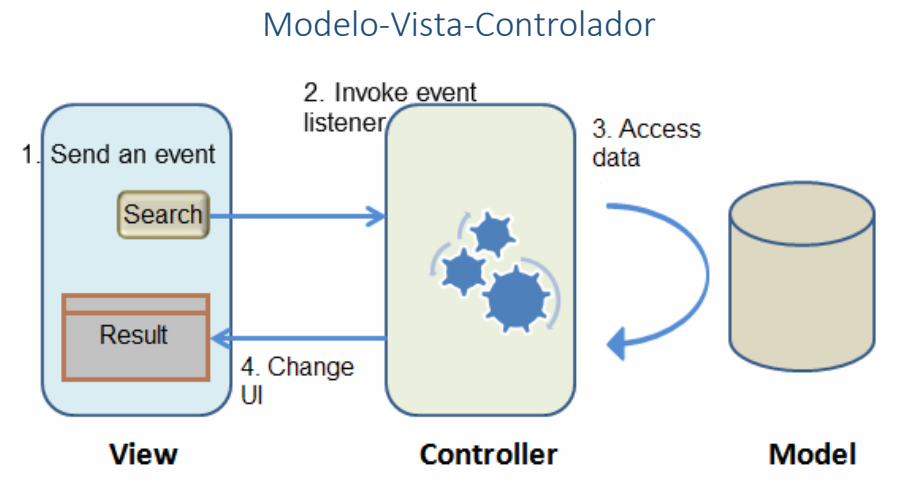


Ilustración 2: Representación del paradigma Modelo-Vista-Controlador.

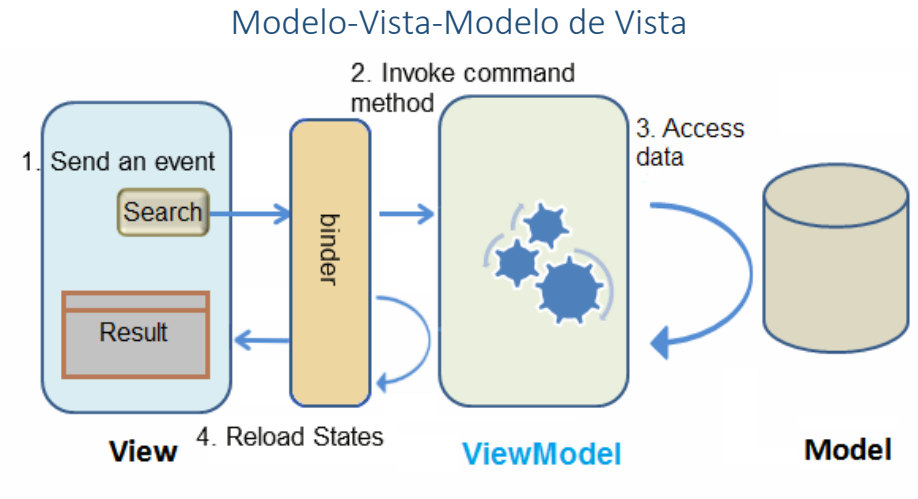


Ilustración 3: Representación del paradigma Modelo-Vista-Modelo de Vista.

Construir una aplicación mediante MVC puede resultar más intuitivo: el programador directamente controla lo que hay en la vista y su comportamiento de forma manual.

Este patrón se caracteriza por tener el control total de los componentes, por lo tanto se pueden crear componentes dinámicamente, controlar componentes propios personalizados o realizar cualquier cosa sobre el componente que este pueda hacer por sí mismo de forma directa y simple.

En la otra cara de la moneda, en el patrón MVVM debido que la capa *ViewModel* esta débilmente ligada con la vista (dicha capa no tiene acceso a los componentes de la vista), podemos usarla con múltiples vistas sin tener que modificarla.

Si la información y comportamiento no cambian, un cambio en la vista no provoca que se tenga que modificar la capa *ViewModel*.

Modelo-Vista-Modelo de Vista

Vistos los puntos fuertes de ambos paradigmas, el modelo escogido para el proyecto fue el patrón MVVM. Los motivos de esta elección recayeron sobre la abstracción que permite la separación de la vista del control de datos de la aplicación.

Como hemos visto anteriormente, la interfaz de usuario de la aplicación puede modificarse sin tocar el código y este no depende de los cambios que se produzcan en la vista.

El modelo puede encapsular la lógica de negocios existente sin temor a sufrir muchos cambios debido a cambios en la vista; El modelo de vista actúa como un adaptador para las clases del modelo y se comporta como un parachoques de los cambios en la vista.

Por otra parte, los desarrolladores pueden crear pruebas unitarias para el modelo de vista y el modelo sin usar la vista y si fuese el caso, diseñadores y desarrolladores pueden trabajar al mismo tiempo y de forma independiente en sus componentes. Los diseñadores pueden centrarse en la vista, mientras que los desarrolladores pueden trabajar en el modelo de vista y los componentes del modelo.

Xamarin Forms

La manera en la que el patrón MVVM se manifiesta en C# y en concreto, en Xamarin forms, es a través de *Data Binding* y *Commands* que serán los encargados de la comunicación entre el Modelo de Vista y la propia Vista.

BINDINGS

Un *Binding* relaciona dos propiedades entre sí de forma que se mantengan sincronizadas. Existen tres tipos de *binding*, “One Way”, “ Two way” y “One Way To Source”.

Con un ***binding* de un sentido**, como su nombre nos indica, siempre que realicemos algún cambio de alguna propiedad en el *ViewModel* se propagará a la vista. En cambio, si el cambio se realiza en la Vista, la propiedad en el *ViewModel* no se actualizará.

Por consiguiente, un ***binding* de doble sentido** sí enviará el valor que se escriba en la Vista a la propiedad del *ViewModel,* además de actualizar la Vista en caso de haber cambios en el *ViewModel.*

Finalmente, un ***binding* de un sentido a la fuente** tendrá un flujo contrario al de un solo sentido, tan solo actualizando el *ViewModel* siempre que se modifique la vista.

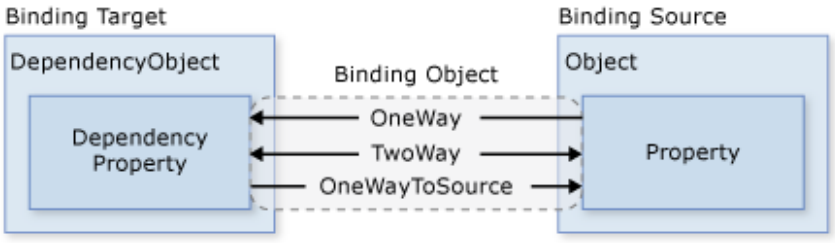


Figura 5: Esquema de los tres tipos de Data Bindings en Xamarin.

A continuación se muestra un ejemplo de un atributo de tipo *String* con un *binding* a un componente *Entry* y las diferencias entre los tres tipos de *Binding.*

```
// Código en nuestro ViewModel:
private string title;
public string Title {
    get {
        return title;
    }
    set {
        title = value;
        RaisePropertyChanged();
    }
}
```

Figura 6: Data Binding en el ViewModel para la propiedad de Título.

```
// Binding en nuestro archivo XMAL
<Entry Text="{Binding Title}" />
<Entry Text="{Binding Title, Mode=TwoWay}" />
<Entry Text="{Binding Title, Mode=OneWayToSource}" />
```

Figura 7: Data Binding en la Vista para la propiedad de Título.

COMMANDS

Los *Commands* son el método que tiene Xamarin para trasladar las acciones o eventos de la vista al *ViewModel*.

A través de la interfaz *ICommand* podemos enlazar un *Command* con el control de una vista a través de un *Binding* de manera similar a como hemos visto anteriormente.

```
// Definición de un Command en nuestro ViewModel
private ICommand _SearchByName;

public ICommand SearchByNameCommand {
    get {
        return _SearchByName ?? (_SearchByName = new Command (
            async () => await ExecuteSearchByNameCommand (),
            CanExecuteSearchByNameCommand));
    }
}

private async Task ExecuteSearchByNameCommand ()
{
    await LoadData (SearchText);
}

private bool CanExecuteSearchByNameCommand()
{
    return SearchText.Lenght > 0;
}
```

Figura 8: Command en una clase de ViewModel para la acción de Buscar en un buscador.

```
// Command binding en nuestro archivo XMAL
<SearchBar Text="{Binding SearchText}"
SearchCommand="{Binding SearchByNameCommand}"></SearchBar>
```

Figura 9: Binding de un Command en el componente SearchBar de la Vista.

4.5.2. Estructura interna

Vista la forma que va a tener nuestra aplicación y sus componentes, podemos tener una idea clara de la estructura y la organización del código de la misma.

A continuación se muestra la estructura del proyecto y la explicación a cada una de las partes:

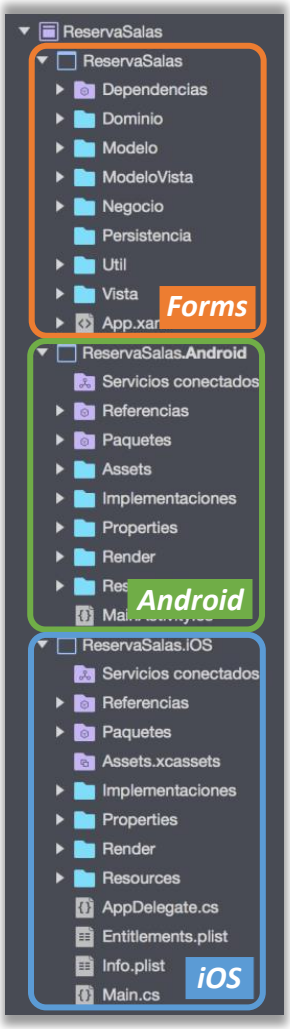


Figura 10: Estructura del proyecto Xamarin.

ReservaSalas: Conforma la parte común de código, escrito en C#, que comparten tanto Android como iOS. En esta sección es donde principalmente se realiza el desarrollo y donde se escribe la lógica de toda la aplicación.

Dentro de este apartado podemos encontrar los siguientes partes de la aplicación:

Dependences: Aquí se encontrarán los *NuGets* y librerías externas que usemos en el proyecto.

Dominio: Donde estarán todas las clases de la aplicación y elementos internos y modelos de datos.

Dentro del *Dominio* podemos encontrar Un sub-apartado de *ServicioWeb*, donde irán los objetos exclusivos de la comunicación con el servidor.

Modelo: Dentro de *Modelo* encontraremos toda la lógica y clases de uso común en la aplicación. Es habitual la creación de clases que se encarguen de trabajar con clases concretas y almacenen toda la lógica relacionada. Dentro de este apartado podría haberse encontrado las clases del *Dominio*, pero se acabó separando por accesibilidad.

ModeloVista: En este apartado encontraremos las clases encargadas de la lógica de las vistas. Son clases que contienen la información de los componentes que el usuario ve en pantalla y se encarga de trabajar con la información. Estas clases suelen trabajar con las clases de Modelo y ayudan a separar la lógica de la vista, ofreciendo más independencia.

Negocio: En negocio encontramos todas las clases encargadas de la comunicación con el servidor. Dichas clases estarán separadas por el tipo de llamadas que realizan, normalmente englobando llamadas sobre un mismo tipo de objeto: *ServicioWebSesion*, *ServicioWebSalas*, por ejemplo.

Este sub apartado también podría encontrarse dentro del apartado del *Modelo* pero se separó por la misma razón que el *Dominio*.

Persistencia: Las clases que traten con bases de datos internas irán aquí. Básicamente, toda la lógica encargada de persistir y organizar diferentes datos de la aplicación. La clase encargada de almacenar los ajustes de la aplicación también la encontraremos aquí.

Util: Como su nombre indica, aquí encontraremos una variedad de clases con diferentes utilidades que nos ayudarán a lo largo de la aplicación.

Dentro de *Útil* podemos encontrar los siguiente sub-apartados:

- ***Behaviors**:* En *Behaviors* encontraremos comportamientos personalizados para diferentes componentes de las vistas. Generalmente nos ayudarán a enlazar diferentes acciones con *Commands* para ejecutar diferente código en respuesta.
- ***Controles**:* Aquí encontraremos diferentes componentes personalizados con un aspecto o comportamiento que se ha tenido que programar de forma expresa.
- ***Converters**:* Clases dedicadas a interpretar y/o convertir datos para las vistas. Normalmente se usan para no hinchar la lógica del *ModeloVista* o realizar conversiones habituales pero que necesitan de una mínima lógica.
- ***Diccionario**:* La localización de la aplicación se encuentra aquí.

ReservaSalas.Android: Este apartado contendrá el proyecto en *Android*. Al trabajar con *Forms*, no interactuaremos mucho con esta parte del proyecto pero será necesario para configurar diferentes aspectos concretos de *Android* o para programar componentes personalizados y su comportamiento en un sistema Android.

ReservaSalas.iOS: Finalmente, aquí encontraremos la parte *iOS* del proyecto. Al igual que ocurre con la parte Android, no interactuaremos mucho con ella gracias a la parte común, aunque será necesario configurar ciertos aspectos del entorno *iOS* de forma directa y concreta además de permisos. También se deberán programar aquí la parte iOS de los componentes personalizados que creemos.

4.6. Diseño de pantallas

Para el diseño de pantallas, se realizaron una serie de *mock-ups* que mostraban los casos de uso del usuario donde se detallaba el comportamiento de los diferentes componentes de las pantallas.

El flujo de la aplicación, detalles puntuales del estilo y otros detalles relacionados con las pantallas y la interacción del usuario también quedan detallados en este documento.

4.6.1. Pantalla *login* y pantalla principal

El diseño de la pantalla de *login* sigue la misma línea de diseño que la pantalla de *login* de otras aplicaciones de la empresa de manera que se mantenga el estilo de las mismas.

En esta pantalla podemos encontrar un enlace a la política de privacidad de la empresa, un botón para acceder a la pantalla de información de la aplicación y las acciones relacionadas con el inicio de sesión: guardar contraseña y aceptar la política de privacidad.

Clicando sobre el botón de *Entrar,* se validará la sesión en el servidor y en caso de ser correcto accederemos a la pantalla principal de la reserva de salas.

En la pantalla principal encontramos tres apartados con diferentes listados: Una lista de reservas actuales del usuario, una lista de salas disponibles en ese momento y una lista de salas disponibles más tarde.

La información que se muestra de las salas o las reservas es la siguiente: una pequeña imagen representativa de la sala reservada o a reservar, el nombre, el edificio donde se ubica, los servicios de los que dispone, su capacidad y en caso de tratarse de una sala a reservar, la hora hasta la que está disponible para su reserva.

Debajo encontramos dos botones: “Reservar sala” y “Mis reservas” que nos llevarán a otras pantallas como detallaremos más adelante.

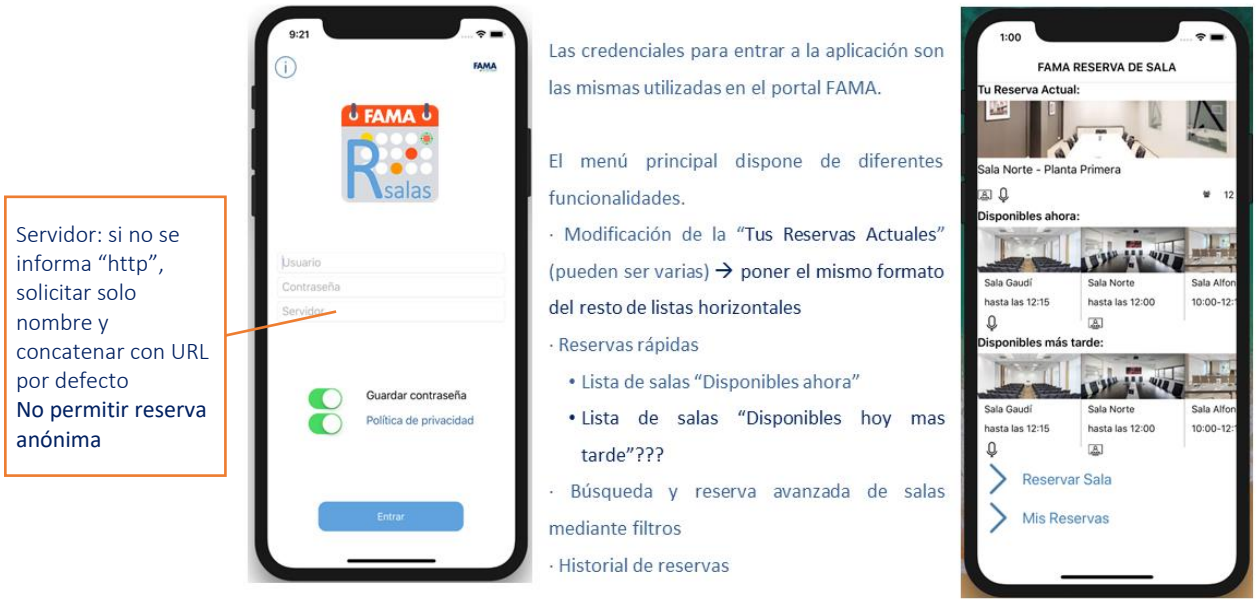


Ilustración 4: Diseño de la pantalla de Login y pantalla principal.

4.6.2. Reserva actual

La pantalla de una reserva se mostrará en diferentes puntos de la aplicación y debe ser genérica en tanto se trate de una sala para reservar, una reserva en curso o una reserva ya finalizada del histórico.

En esta pantalla se mostrará la siguiente información de una sala: Una imagen representativa, su nombre y ubicación, los servicios de los que dispone, la capacidad, el asunto de la reserva si se trata de una reserva y finalmente un *grid* de botones con la franja horaria de reserva de la sala.

Dicho *grid* se encontrará bloqueado con una franja horaria marcada en el caso de tratarse de una reserva. De otra forma, el *grid* estará desbloqueado y permitirá

seleccionar una franja horaria de reserva al usuario. De la misma manera, las horas que no estén disponibles para reservar se mostrarán en gris y no se permitirá al usuario seleccionarlas.

En concreto, llegaremos a la pantalla de una reserva actual desde la primera lista del menú principal y se mostrará la información de una reserva de sala realizada por el usuario.

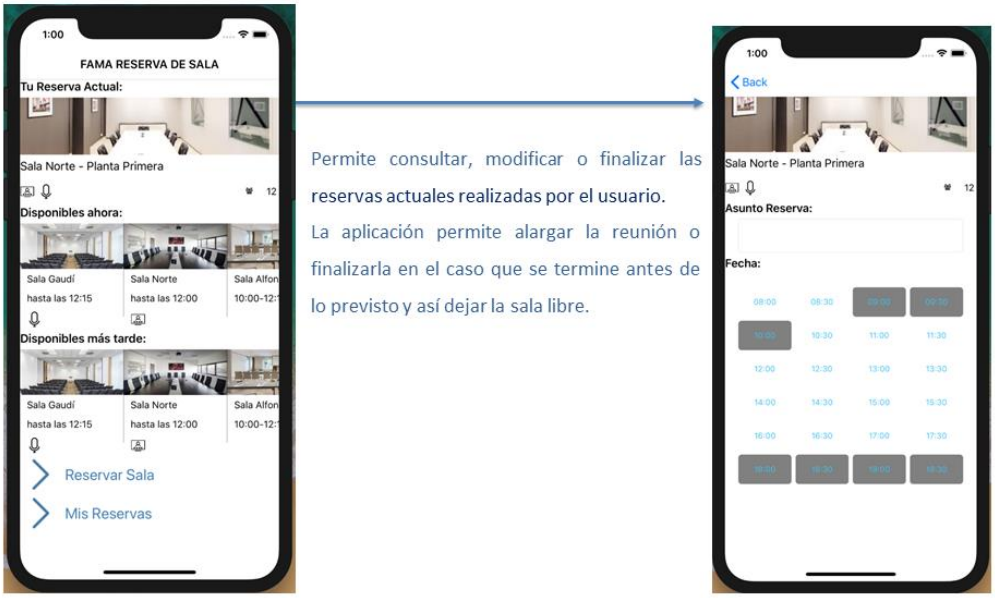


Ilustración 5: Diseño de la pantalla de principal, pantalla de una reserva y el flujo.

4.6.3. Reserva directa

Llegaremos a la pantalla de reserva mediante las dos listas de salas disponibles para reservar del menú principal. En esta pantalla se nos mostrará la información de una sala disponible para reservar con las franjas horarias libres para su reserva.

La demás información mostrada será la especificada en el apartado anterior. En esta pantalla el usuario podrá especificar una franja horaria disponible de la sala para reservar además de un asunto que represente el motivo de la reserva.

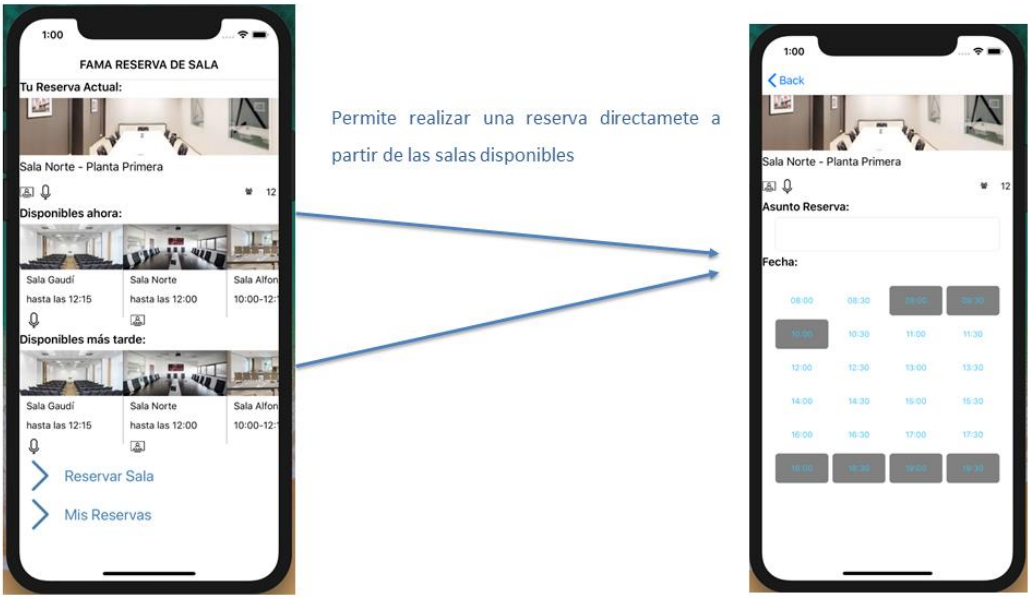


Ilustración 6: Diseño de la pantalla de principal, pantalla de una reserva y el flujo.

4.6.4. Reserva mediante buscador

La reserva mediante buscador resulta en la funcionalidad más compleja de la aplicación, la que requiere de más pasos y la que cuenta con más funcionalidades.

Accederemos a la pantalla de búsqueda de salas desde la primera opción “Reservar Sala”, debajo de las listas de reservas y salas disponibles del menú principal.

La primera pantalla mostrada será una pantalla con diferentes aspectos de una sala que el usuario deberá o no especificar según el criterio de búsqueda que requiera.

Las principales características por las que se permite filtrar al usuario son: El inmueble donde se ubica la sala, la fecha de la reserva, la duración de la misma, la franja en la que se quiere realizar la reserva y el número de asistentes.

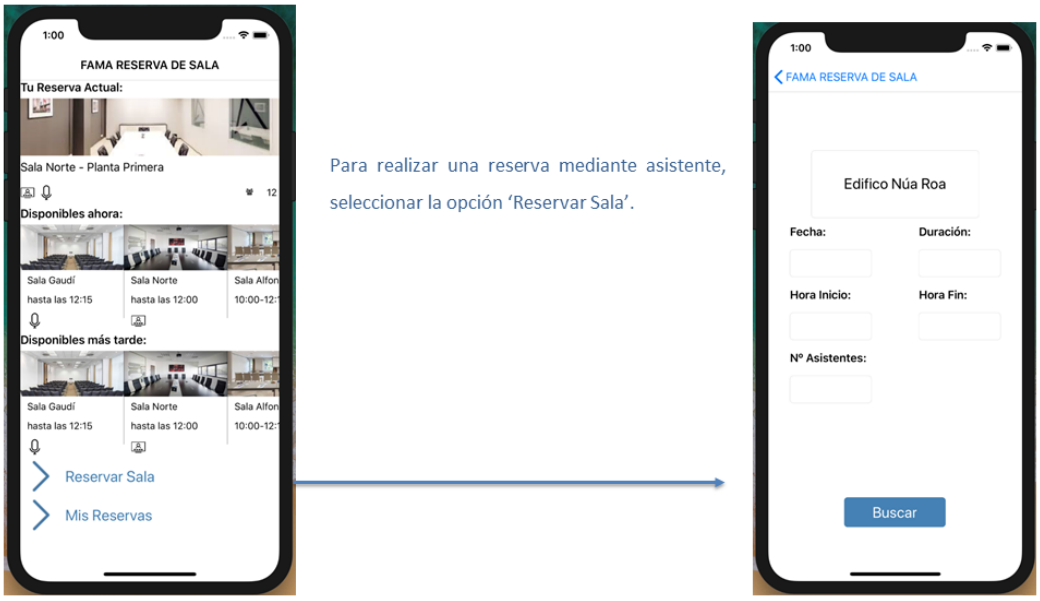


Ilustración 7: Diseño de la pantalla de principal, pantalla de búsqueda y el flujo.

Uno de los criterios por los que el usuario puede filtrar las salas es el inmueble donde se ubica la sala. Esto resulta así pues, un cliente puede contar con diferentes edificios y/o espacios dentro de un complejo o puede estar compuesto por diferentes complejos con salas y espacios a reservar.

Al abrir esta pantalla, el inmueble por defecto vendrá definido por el espacio donde se ubica el usuario de la aplicación. Este parámetro estará definido en el servidor y se obtendrá en el momento del inicio de sesión.

El usuario podrá cambiar el espacio clicando sobre el nombre en la pantalla de filtros. La aplicación cargará una lista de espacios disponibles ordenados por cercanía que el usuario podrá seleccionar. La distancia se calculará mediante la ubicación GPS.

Ilustración 8: Diseño de la pantalla de búsqueda, lista de edificios y el flujo.

Una vez el usuario haya definido los criterios deseados, podrá clicar sobre el botón *Buscar* situado en la parte inferior de la pantalla; la aplicación lanzará la petición al servidor con la búsqueda de salas y el usuario podrá acceder a una nueva pantalla con la lista de salas que cumplan dichos criterios.

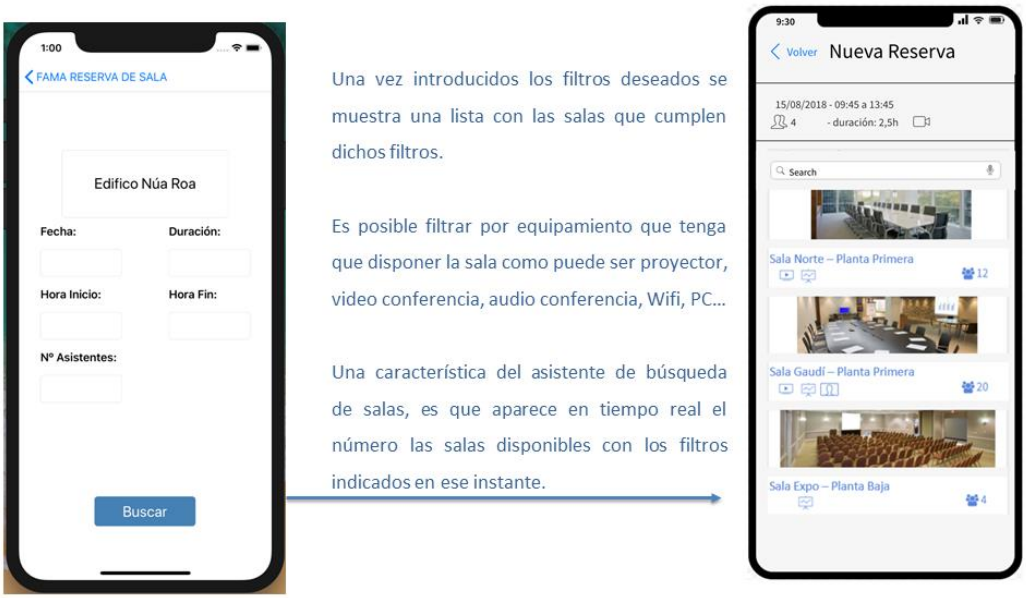


Ilustración 9: Diseño de la pantalla de búsqueda, pantalla de lista de salas filtradas y el flujo.

Finalmente, desde la lista de salas filtradas se podrá acceder a su reserva. El flujo será similar al especificado en el punto de *Reserva directa*: El usuario accederá a una pantalla con las características de la sala donde se verá su disponibilidad y podrá seleccionar una franja horaria disponible para reservar y un especificar un asunto.

Ilustración 10: Diseño de lista de salas filtradas, pantalla de una reserva y el flujo.

4.6.5. Historial de reservas

Otro de los aspectos que ofrece la aplicación es la de poder consultar las reservas realizadas por el usuario. Esta opción se encuentra en la pantalla principal junto al botón de *Reservar Sala* del buscador.

Esta opción englobará tanto las reservas pendientes como el histórico en una pantalla con una "tabbed page". Este diseño fue elegido por la similitud de la información mostrada y por la necesidad de separarla de forma clara dentro del mismo marco.

Ilustración 11: Diseño de la pantalla de principal, pantalla historial de reservas y el flujo.

La primera página a la que se accederá a través del menú principal será la lista de Reservas pendientes. En esta página el usuario podrá ver las reservas que haya realizado que estén en transcurso o con la fecha de inicio superior a la fecha actual.

Clicando sobre cualquiera de las reservas se podrá acceder a la información de la reserva como se detalla en la reserva actual. Si se trata de una reserva pendiente es posible modificarla o anularla.

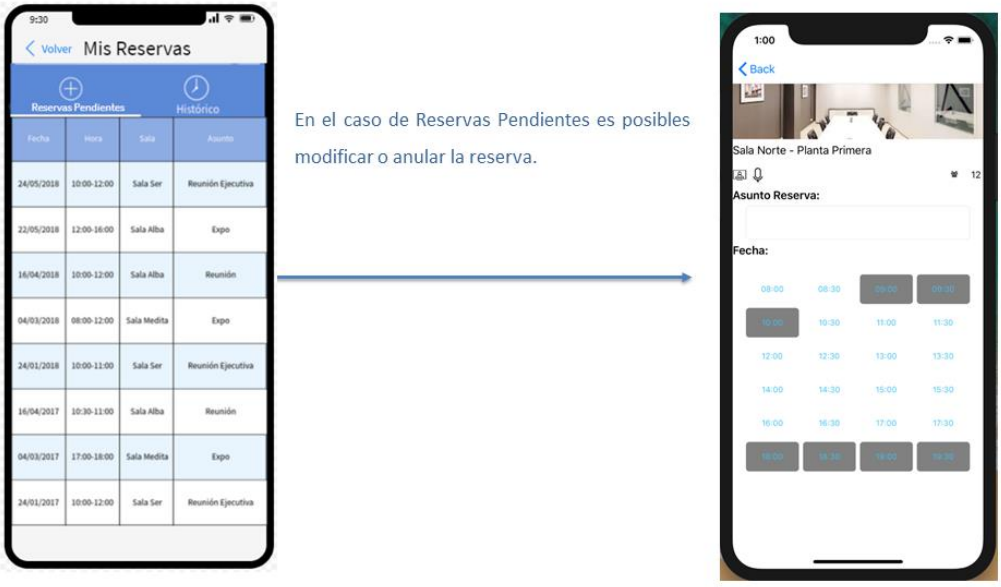


Ilustración 12: Diseño de reservas pendientes, pantalla de una reserva y el flujo.

En la otra pestaña, se encontrará el histórico de reservas. En este listado de reservas se mostrarán aquellas que ya hayan finalizado, más concretamente, las que su fecha de finalización sea anterior a la fecha actual.

Clicando sobre cualquiera de las reservas se podrá acceder a la información de la reserva como se detalla en la reserva actual. La información mostrada no será editable y tan solo será de consulta.

Ilustración 13: Diseño de la pantalla de histórico de reservas, pantalla de una reserva y el flujo.

5. Desarrollo

Una vez realizado el Análisis funcional, la especificación de los requisitos y habiendo definido un esquema para la APP, ya se pueden dar los primeros pasos en el desarrollo del proyecto.

En esta sección veremos el plan de acción seguido en el desarrollo, desgranaremos el desarrollo en sí tanto de la APP como de la Aplicación Web, nos detendremos en las incidencias que surgieron en el transcurso del mismo y finalmente hablaremos del futuro del proyecto y el margen de mejora y ampliación de la aplicación.

5.1. Plan de Acción

Como hemos comentado en apartados anteriores, ya han existido anteriormente, dentro de la empresa, otros proyectos dentro del mismo marco, aunque con otros objetivos.

En concreto, existe otro desarrollo que marcó las bases en la creación de aplicaciones con Xamarin.

Debido a la inexperiencia en el entorno y con el objetivo de realizar una maqueta de la aplicación en el menor tiempo posible, antes de empezar a generar código fue conveniente estudiarlo, ver su estructura y detectar posibles partes comunes que pudiesen tener ambos proyectos.

A demás, dentro de la planificación temporal vista en el apartado 3.4.3, hubo cambios en la fase 3 debidos a la especificación del problema inicial y su desarrollo.

En resumen, la confección de una maqueta ganó importancia a la par que el aprendizaje de Xamarin, que se extendió parte de la creación de la maqueta. Por otro lado, ciertos puntos de la fase 3, como el diseño de las pantallas y navegación, cobertura legal y pruebas y evaluación se extendieron de forma transversal en el desarrollo en los diferentes casos de uso debido a la metodología de trabajo propuesta.

Dichos cambios nos llevaron a modificar la planificación y extender y detallar la fase de desarrollo.

La planificación resultante a los cambios fue la siguiente:

Tarea	*Tiempo estimado (días)*	*Tiempo estimado (horas)*
Especificación de sistema y los servicios	5 días	25 h
Diseño de las pantallas y navegación	1 día	5 h
Especificaciones de seguridad	3 días	15 h
Cobertura legal	1 día	5 h
Total fase 3: Fase de desarrollo: Plan de Acción	**10 días**	**50 h**

Total fase 3: Fase de desarrollo: Maqueta	**11 días**	**55 h**
UC Login + UC Logout + UC Acerca de	1 día	5 h
UC Pantalla Inicio + UC Menú principal	5 días	25 h
UC Nueva Reserva	2 días	10 h
WS Login + WS Test de conectividad	2 días	10 h
WS Salas disponibles + WS Reservas Usuario	4 días	20 h
WS Alta Reserva de Sala	2 días	10 h
Publicación Play Store Versión Fase 1	1 día	5 h
Publicación App Store Versión Fase 1	1 día	5 h
Total fase 3: Fase de desarrollo: Fase 1	**18 días**	**90 h**
UC Buscar Salas	3 días	15 h
UC Histórico Reservas	2 días	10 h
UC Modificar Reserva	4 días	20 h
UC Anular Reserva	1 día	5 h
WS Buscar Salas	1 día	5 h
WS Modificar Sala + WS Anular Reserva	3 días	15 h
WS Descargar Documento	1 día	5 h
Ajustes App	5 días	25 h
Publicación Play Store Versión Fase 2	2 días	10 h
Publicación App Store Versión Fase 2	2 día	10 h
Documentación	5 días	25 h
Total fase 3: Fase de desarrollo: Fase 2	**24 días**	**120 h**

Tabla 10: Planificación final del proyecto con distinción de tareas y su coste en días y horas.

Puntualizar que, aunque el diseño de pantallas y navegación y la cobertura legal se realizaron en múltiples puntos del proyecto si fue necesario realizarlos de forma previa al desarrollo en sí en la confección del Plan de Acción como punto de partida para el mismo.

5.2. Desarrollo

El desarrollo en sí siguió la secuencialidad del plan de acción aunque las horas de dedicación diarias fueron superiores a las planeadas.

Una vez pasada la fase de análisis y estudio de *Xamarin* y la confección de la maqueta, las horas de dedicación diaria aumentaron debido al volumen de trabajo que, aunque repartido, requería de más tiempo para adecuarse a los plazos estipulados en el plan de acción.

La fase de análisis de requisitos concluyó con la confección de diversos documentos concisos referentes al análisis funcional y diseño, el esquema de la App, y la revisión del diseño de pantallas y la planificación.

Estos documentos, aunque completos, resultaron concisos y más simples que el análisis realizado en este documento, más cercanos a un desarrollo ágil; fueron creados con la idea de ser ampliados y mantenidos y fueron mutando paralelamente al desarrollo.

Xamarin y APP Solicitudes

Ya de forma temprana, las horas de estudio de Xamarin y de análisis de la aplicación anterior se extendieron más de lo previsto. La aplicación anterior, aunque ya conformaba un proyecto de una aplicación finalizada, seguía una estructura y una metodología que no se adecuaban a este.

Las partes comunes se limitaron en la pantalla de *inicio de sesión*, la pantalla de *acerca de*, la localización y algunas llamadas a servidor.

Con todo y con esto, todas las partes nombradas tuvieron que ser rehechas de nuevo por diversos motivos:

La **página de *inicio de sesión*** no se adaptaba bien a pantallas pequeñas, con algunos solapamientos y desaparición de algunos componentes.

Aunque el comportamiento de los componentes era *responsive,* hubo que rediseñar la pantalla para evitar que, en pantallas pequeñas, los *switch* de “recordar contraseña” y “aceptar política de privacidad” desapareciesen. A demás, se cambió la posición y lógica del icono de la aplicación para que no se desplazase mucho en pantallas grandes.

La **pantalla *acerca de*** era confusa y se rediseñó para hacerla más clara. La idea original era disponer los campos de manera horizontal, pero la información más larga se aglutinaba. Se dispusieron los campos de forma vertical repartidos por el espacio.

Ilustración 14: Derecha, Pantalla "Acerca de" de la aplicación de Solicitudes. Izquierda, Pantalla "Acerca de" de la aplicación Reserva de Salas.

La **localización** de la aplicación, aunque estaba hecha mediante los recursos estándares ofrecidos por Xamarin, estaba incompleta e implementada de una forma que no era adecuada. Los diccionarios de recursos estaban duplicados y sin entrar en aspecto más concretos, se simplificó y optimizó.

Finalmente, de las **llamadas a servidor**, aunque algunas de ellas eran utilizadas en la aplicación de reserva de salas, hubieron de ser optimizadas para implementar paralelismo y usar de forma avanzada los *Task* de C#.

Diseño de pantallas

El diseño de pantallas, aunque partía de los *mock-ups*, fue variando durante el desarrollo y fue adaptándose a los componentes de *Xamarin*.

Esto principalmente fue debido a la inexperiencia y el desconocimiento del equipo desarrollador sobre *Xamarin*, aunque también era una de las partes con las que contaba libertad en el desarrollo y en las que tuve que tomar más decisiones bajo mi propio conocimiento.

Del mismo modo, algunas pantallas causaron ciertos problemas por la falta de componentes en *Xamarin* o por la compleja adaptación de algunos a la multiplataforma; Ciertas partes de la interfaz no se visualizaban bien en Android o iOS, provocando que fuese necesaria la búsqueda de alternativas a ciertos componentes o funcionalidades que ya estaban fijadas.

Muchos de los inconvenientes venían marcados por la densidad de píxeles en pantalla de los dispositivos. Habitualmente, los dispositivos con iOS tiene una densidad de píxeles mayor que la mayoría de los dispositivos con sistema Android y a la hora de visualizar la aplicación esto causa que ciertas pantallas no se vean de la misma manera o que se pierda información en pantallas con una densidad de píxeles baja.

Por otra parte, algunas pantallas debieron ser cambiadas para simplificar ciertas funcionalidades y ganar en accesibilidad, sencillez y usabilidad:

En **la pantalla principal**, la lista de "Salas disponibles más tarde" fue eliminada por no definir de forma clara un comportamiento sencillo y útil; Su funcionalidad podía ser cubierta por la primera lista, "Salas disponibles ahora", y por el buscador, en el caso de Salas disponibles en el mismo día y fuera del rango de la lista de Salas anterior.

La **pantalla de búsqueda de salas** fue rediseñada íntegramente en su desarrollo, haciendo uso de diferentes componentes nativos de Xamarin y dejando a un lado entradas de texto, poco intuitivas, toscas y complejas para indicar valores que no fueran texto. Los componentes candidatos fueron *pickers*, *sliders* y *sttepers.*

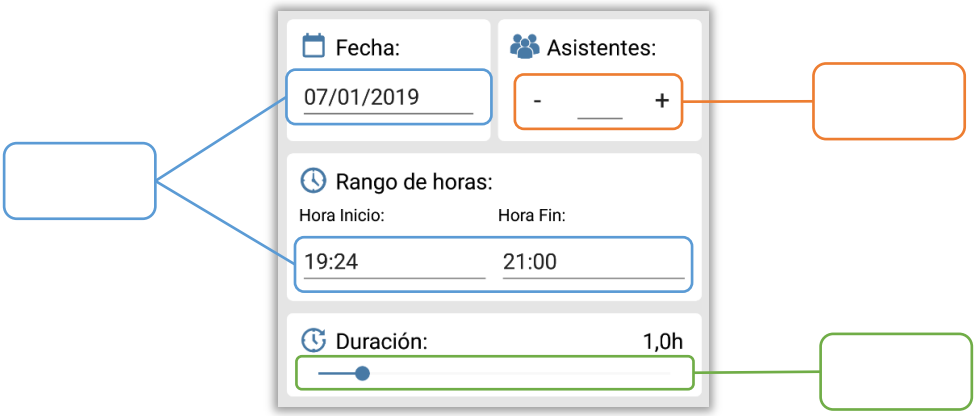


Ilustración 15: Fragmento de la pantalla de búsqueda de salas con distintos componentes.

Componentes complejos

Durante el desarrollo de la pantalla *de reserva de una sala*, uno de los componentes que causó problemas debido a su complejidad fue el *grid* de franjas horarias disponibles de la salas.

El comportamiento del *grid* fue rediseñado en varios puntos del desarrollo y al final se decidió por un control semi-automático de selección de franjas horarias:

Por una parte, el grid solo mostrará, en caso de reserva, las franjas horarias disponibles para reservar, es decir, las filas de horas que sean anteriores a la hora actual no se mostrarán.

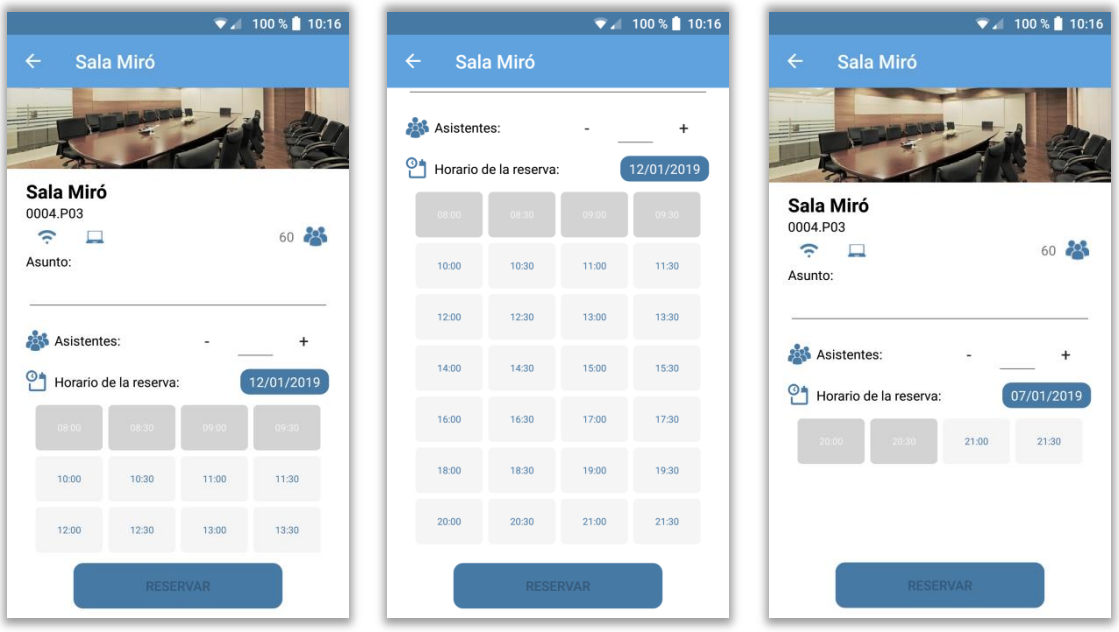


Ilustración 16: Pantalla de Reserva de una Sala. Diferencia del grid de horas a las 9:00 de la mañana (izquierda y centro) y a las 20:00 de la tarde (derecha).

Por otra parte, la selección de una franja horaria se realizará con la selección de las fechas de inicio y final de la reserva:

La primera hora seleccionada, marcará la fecha de inicio. La segunda hora seleccionada marcará la fecha final de la reserva y, automáticamente, se seleccionarán todas las

horas intermedias. En el caso que la segunda hora seleccionada sea menor que la primera, el primer clic denotará la hora final y el primero, la inicial de la reserva.

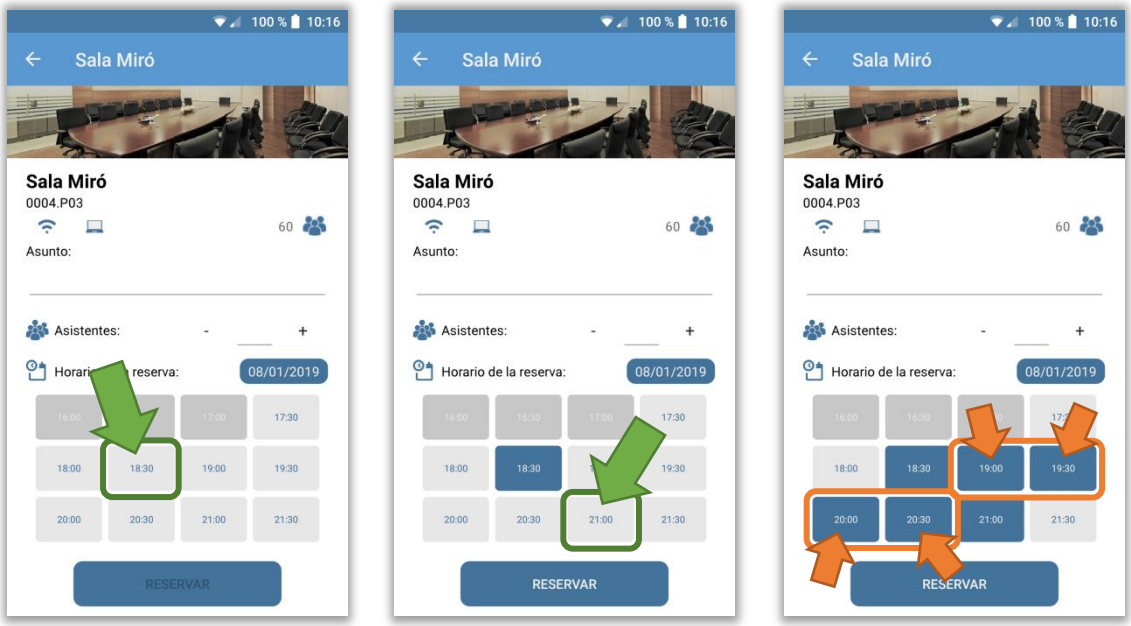


Ilustración 17: Pantalla de Reserva de una Sala y comportamiento del grid de la franja horaria.

Del mismo modo, si entre la primera fecha seleccionada y la segunda existe alguna hora bloqueada, se seleccionarán, empezando desde la primera fecha marcada, todas las horas posible hasta la fecha bloqueada.

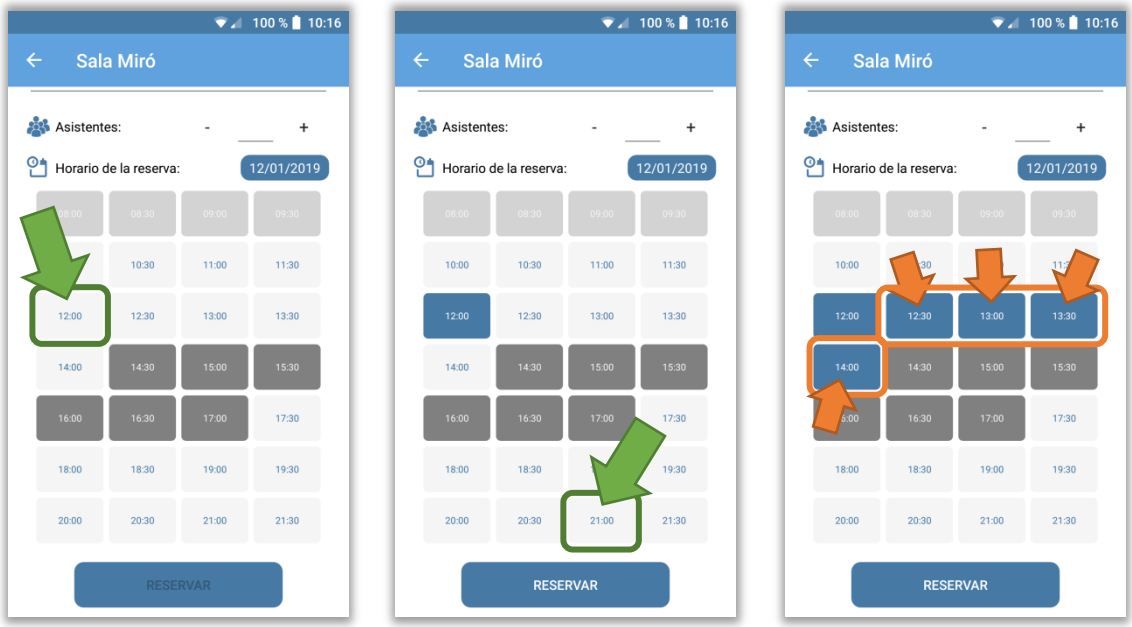


Ilustración 18: Pantalla de Reserva de una Sala y comportamiento del grid de la franja horaria con horas bloqueadas en medio.

Como podemos ver, el *grid* de botones es un componente con cierta complejidad que requiere de un control personalizado y hubo que integrarlo con la metodología de Modelo-Vista-Modelo de Vista, lo que comportó más trabajo del esperado.

Servicios web y servidor

Una vez las pantallas ya estuvieron funcionando de una manera óptima, se empezó a desarrollar los servicios web que iban a usar dichas pantallas.

Con la aplicación anterior, se creó toda la lógica de servidor y se definió la comunicación App-Aplicación web por lo que el primer paso trató en ver como se había resuelto.

En esencia, se usó el paradigma REST junto a objetos JSON, lo que permite transportar objetos con muy pocas limitaciones mediante una representación de la información aplanada mediante un conjunto de parejas de elementos *Tag-Valor* que es interpretada en el servidor y viceversa de una forma óptima.

```
{
    "ID_DOCUMENTO":3010010,
    "ASUNTO":null,
    "FECHA_INICIO":"08/01/2019 08:00:00",
    "FECHA_FIN":"08/01/2019 15:00:00",
    "ID_SALA":3000020,
    "ID_SALARESERVA":4000045,
    "NOMBRE_SALA":"Sala Miró",
    "NOMBRE_ESPACIO":"Edificio Trade Norte",
    "SERVICIOS_SALA":
        3,
        4,
        5
    ],
    "ASISTENTES":4,
    "CAPACIDAD":15,
    "FECHA":"08/01/2019 08:00:00"
}
```

Figura 10: Representación de un objeto "Sala" en formato JSon.

Sin embargo, en la parte servidor, los servicios de la aplicación, por el hecho de no haber coexistido con ninguna otra App, no se encontraban organizados de una manera que permitiera su escalabilidad de forma natural y hubo que buscar la manera de hacerlo y organizar el código existente sin que la App de Solicitudes dejase de funcionar.

Otro aspecto a destacar es que, durante la programación de los servicios, se decidió simplificar las llamadas, unificando la llamada de obtención de salas y distinguiendo su naturaleza mediante valores y filtros que el servidor usa para la consulta en la base de datos. Se hizo de esta manera ya que desde el servidor la funcionalidad es la misma y se evita duplicidad de código y disponemos de la libertad para hacerlo al conocer y diseñar ambos lados de la comunicación.

Cobertura legal y "NuGets"

Aunque ciertos aspectos a nivel legal son mirados de manera previa al desarrollo (lenguajes, herramientas, etc…), durante el desarrollo también se tuvo que tener presente debido a los *NuGets.*

Los *NuGets* en C#, son librerías de código (parecido a las *Gems* de *Rubí,* por ejemplo) que implementan diversas funcionalidades y son muy usados en Xamarin. Como los definen en la propia página de Microsoft, "*los paquetes de NuGet son unidades de código reutilizable que otros desarrolladores ponen a su disposición para que los use en sus proyectos*[32]".

Microsoft cuenta con una "galería" de *NuGets* que son fácilmente obtenibles desde Visual Studio que comprenden desde componentes nuevos a funcionalidades nuevas dentro del *framework* de Xamarin. Estos *NuGets* cuentan con diferentes licencias y, aunque la mayoría cuentan con una licencia del tipo *MIT* de carácter libre[33], los hay con licencias que requieren la adquisición de una licencia tanto para su uso en una compañía como a nivel de comercialización.

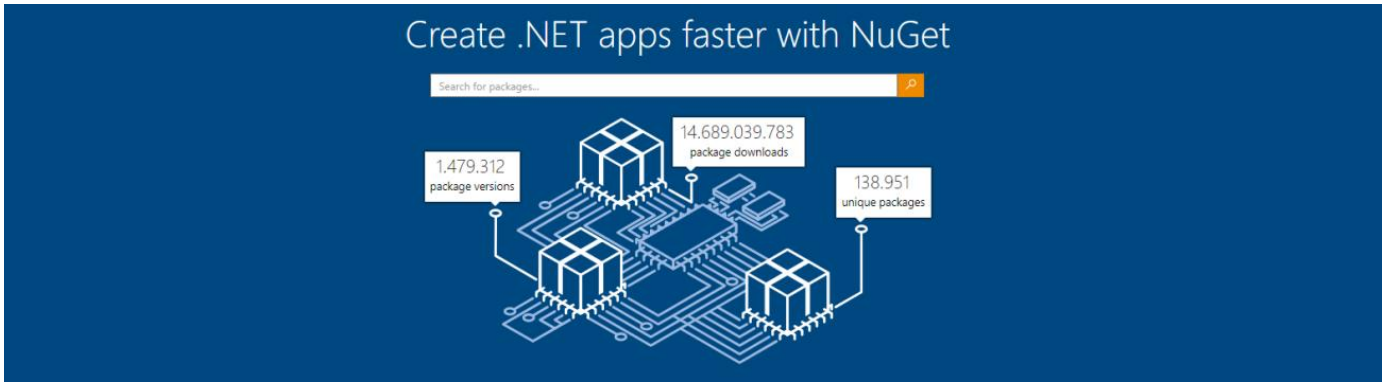


Ilustración 19: Portada de la galería de NuGets de Microsoft.

En este punto, nos encontramos que, aunque Xamarin es un un *framework* completo, el uso de *Nugets* aumenta el alcance a nuevas funcionalidades y ayuda a completar dicho entorno con código proporcionado por la comunidad e incluso equipos profesionales dentro de Microsoft.

En concreto, la aplicación de Reserva de Salas hace uso de diversos *NuGets* de carácter libre para hacer uso de componentes personalizados y más complejos que los estándar, funcionalidades como la recoloración de iconos de forma sencilla e incluso una librería para el manejo de objetos JSON y la comunicación REST.

A continuación veremos la lista de *NuGets* que utiliza la aplicación y una pequeña descripción. En los *Anexos* se han incluido los códigos de ejemplo de los más significativos y los más útiles.

Acr.UserDialogs[34]***:*** *NuGet* que facilita el mostrar mensajes por pantalla. Podremos fácilmente crear alertas, diálogos, *pickers y demás* de una forma sencilla y vistosa. Cuenta con una licencia MIT.

Newtonsoft.Json[35]***:*** Potente librería para el tratado y conversión de objetos JSon en C#. Cuenta con una licencia MIT.

[32] "Inicio rápido: Instalación y uso de un paquete en Visual Studio – Microsoft" https://docs.microsoft.com/es-es/nuget/quickstart/install-and-use-a-package-in-visual-studio
[33] "The MIT License" https://opensource.org/licenses/MIT
[34] "UserDialogs – aritchie" https://github.com/aritchie/userdialogs

Plungin.CrossPlatformTintedImage[36]***:*** Componente personalizado que nos permitirá tintar imágenes para evitar tener duplicada la misma imagen con diferentes colores en nuestro proyecto. Cuenta con una licencia MIT.

Plungin.Pemissions[37]***:*** Sencillo *NuGet* que nos permite comprobar los permisos del usuario en cualquier plataforma y pedirlos al usuario en el caso de ser necesario. Muy útil para el acceso al GPS, la cámara y la galería. Cuenta con una licencia MIT.

Refractored.XamForms.PullToRefresh[38]***:*** Componente personalizado que nos permitirá recargar una página que permita *scroll* como si de una lista se tratase. Cuenta con la misma animación de recarga. Cuenta con una licencia MIT.

Xam.Plungin.Media[39]***:*** Librería que nos facilita las peticiones de contenido al usuario como imágenes, vídeos y demás. Muy usado para solicitar imágenes al usuario a través de la cámara o la galería. Cuenta con una licencia MIT.

Xam.Plungins.Forms.ImageCircle[40]***:*** Otro componente personalizado que nos permite tener imágenes con forma circular. Algo que no nos permite de forma natural el componente básico de *Xamarin*. Cuenta con una licencia MIT.

Xam.Plungin.Settings[41]***:*** Con esta librería contaremos con unas funcionalidades sencillas y directas para generar una clase que almacene las preferencias y ajustes de la aplicación desde cualquier parte y dispositivo. No hace uso de bases de datos por lo que ahorramos en espacio y ganamos en sencillez a dicha alternativa.

Xamarin.Essentials[42]***:*** Este *NuGet* es el resultado de muchos otros *NuGets* populares y básicos distribuidos de forma oficial por Microsoft.

Conforma todo un seguido de funcionalidades muy útiles tales como librerías para obtener información del dispositivo, funcionalidades para el copiado de contenido en el *clippboard,* funciones de geolocalización o control de la vibración entre muchos otros. Cuenta con una licencia MIT. Cuenta con una licencia MIT.

Xamarin.Forms: *NuGet* básico utilizado por el Sistema y por el propio proyecto *Xamarin.Forms.*

---

[35] "Newtonsoft. Json – Newtonsoft" https://www.newtonsoft.com/json
[36] "Tinted Image – shrutinambiar" https://github.com/shrutinambiar/xamarin-forms-tinted-image
[37] "Permissions Plungin – James Motemagno" https://github.com/jamesmontemagno/PermissionsPlugin
[38] "PullToRefresh Layout – James Montemagno" https://github.com/jamesmontemagno/Xamarin.Forms-PullToRefreshLayout
[39] "Media Plungin – James Montemagno" https://github.com/jamesmontemagno/MediaPlugin
[40] "ImageCircle – James Montemagno" https://github.com/jamesmontemagno/ImageCirclePlugin
[41] "Settings Plungin – James Montemagno" https://github.com/jamesmontemagno/SettingsPlugin
[42] "Essentials Plungin – Microsoft" https://docs.microsoft.com/es-es/xamarin/essentials/

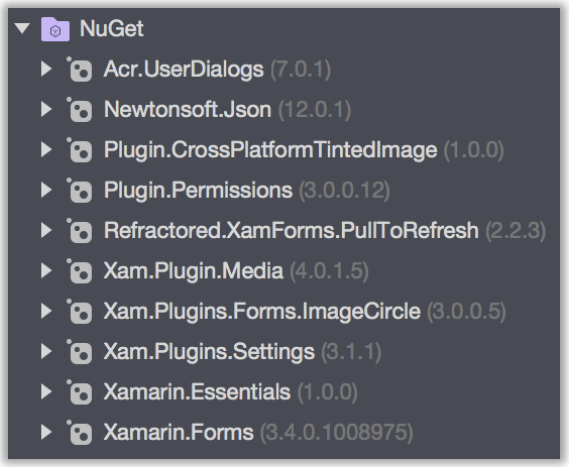


Figura 11: Listado de NuGets del proyecto.

Publicación de la aplicación

Aunque comentaremos aspectos concretos en futuros apartados, la publicación de la aplicación fue un proceso complejo que nos trajo muchos quebraderos de cabeza.

Uno de los objetivos iniciales del proyecto era el poder hacer una aplicación multiplataforma, por lo que una vez realizado el proyecto, las aplicaciones debían estar disponibles tanto en *Play Store* de *Google* como en la *App Store* de *Apple.*

En concreto, se publicaron en las tiendas dos veces; la primera vez que se publicaron fue al finalizar la primera fase, con una versión donde poder ver a grandes rasgos las funcionalidades de la aplicación a modo de "presentación" y la segunda y última dentro del alcance de este proyecto, al terminar la segunda fase, con ya todo el desarrollo terminado y con la aplicación lista para usarse.

Previamente a estas dos publicaciones, realizamos una publicación adicional una semana anterior, una versión beta, para poder distribuir la aplicación a una red de *testers* y así de esta manera, poder tener un *feedback* temprano *y realizar los últimos ajustes* antes del lanzamiento de las aplicaciones.

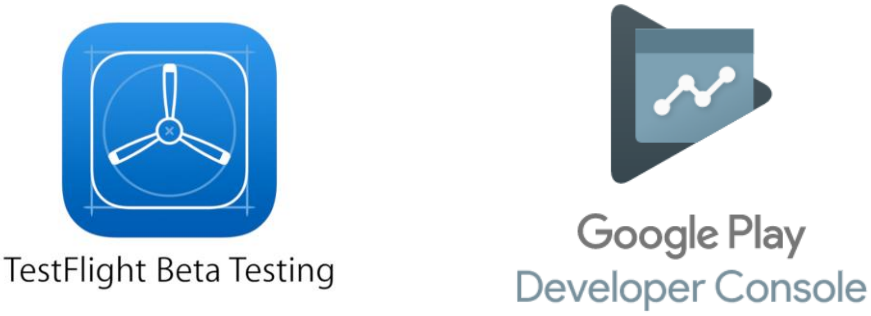


Figura 12: TestFlight y Play Console son la respuesta a la gestión de pruebas de aplicaciones para desarrolladores de Apple y Google respectivamente.

El aspecto a destacar de esta parte del proyecto resulta en la complejidad del proceso de publicación y definición de la App en el entorno de *Apple* en comparación con el de Google. Esta diferencia la encontramos desde la configuración de la cuenta para poder desarrollar, el registro de dispositivos concretos para poder probar la

aplicación en iOS y, sobre todo, en el filtro y la prueba manual que ha de pasar la aplicación antes de ser publicada.

Figura 13: Simplificación de los procesos de publicación de una App.
A la izquierda, el proceso iOS. A la derecha, el proceso Android.

Compilación y firma de aplicaciones

Como podemos ver de forma simplificada en la figura 12 del apartado anterior, dentro de los procesos de producción y publicación de las aplicaciones, se tuvieron que hacer muchos pasos previos fuera del desarrollo de la aplicación en sí.

Uno de estos procesos fue la obtención de las licencias de certificados para poder desarrollar y firmar aplicaciones y, en el caso de iOS, generar la licencia de distribución y el registro de dispositivos para pruebas.

Cabe mencionar que, de anteriores proyectos, las claves estaban desorganizadas y había duplicidad de ellas, y fue necesario una redistribución y creación de claves nuevas para simplificar el proceso.

Aspectos generales

Una vez vistos todos los aspectos del desarrollo, y antes de entrar más a fondo en las incidencias, veremos un par de aspectos más a destacar.

Al finalizar la primera fase, se realizó una reunión a nivel comercial donde se mostró la aplicación a los demás trabajadores de la empresa. En esta reunión se presentaron diferentes aplicaciones que se querían realizar en un futuro y la hoja de ruta que iba a tomar el desarrollo de aplicaciones.

En ese punto, cobró mucha más importancia una buena base de desarrollo en esta aplicación y un desarrollo con visión de futuro, con una programación entendible y escalable.

Factores como, la organización del código de la aplicación y la distribución de los servicios web en el servidor se vuelven importantes y vitales para un buen mantenimiento y para el trabajo en común para con otros desarrolladores.

Por otra parte, debido a la fase de desarrollo de la maqueta y la continua evolución de las pantallas, varias funcionalidades programadas en la segunda fase se realizaron en la primera. No obstante, las llamadas a servidor se implementaron dentro de su programación.

5.3. Incidencias y contratiempos

En este apartado desgranaremos las incidencias más notables del desarrollo y las expondremos de una manera más extensa. Cabe destacar que todas las incidencias encontradas responden a un perfil técnico relacionado con el entorno y *framework* de *Xamarin*.

Escalabilidad del código

Como ya hemos comentado en apartados anteriores, el desarrollo de la aplicación anterior tenía como objetivo tratar con tecnologías nuevas y dar los primeros pasos en *Xamarin* y el desarrollo de aplicaciones multiplataforma.

De esta forma, y tratándose este proyecto de el posterior a el primero, se ha heredado un proyecto que, aunque era consciente de proyectos siguientes, no puso toda su atención en la escalabilidad y en la convivencia con otras aplicaciones.

Por este motivo una parte de este proyecto consistió en *refactorizar* parte del código existente y adaptarlo.

Para la parte servidor, se organizaron las llamadas por tipología, es decir, las peticiones de salas, por ejemplo, irían agrupadas y separadas de espacios facilitando su distinción y su accesibilidad. Anteriormente las llamadas estaban organizadas por aplicaciones, dando problemas para conocer que peticiones están implementadas y cuáles no, generando así solapamientos y cierta dificultando para encontrar dichas llamadas.

Limitaciones de Xamarin.Forms

Uno de los problemas habituales que me he encontrado durante el desarrollo de la aplicación ha consistido en la falta de respuestas estándares en *Xamarin.Forms* para funcionalidades complejas o funcionalidades que se alejan de lo establecido dentro de *Xamarin*.

Xamarin.Forms es un potente *framework* que consigue compilar de forma nativa de una manera correcta y eficiente, pero a cambio hace un sacrificio en funcionalidad. Al tratarse de una conversión multiplataforma, nos encontramos en que no todos los componentes tienen su correspondencia en alguno de los entornos, que es la propuesta de *Xamarin.Forms.*

Este hecho nos expone a situaciones ciertamente desfavorables cuando el diseño está creado pensando en una funcionalidad en concreto o un comportamiento basado en alguna plataforma. Esto no significa que haya diseños que no se puedan realizar en *Xamarin,* pero el trabajo de programación se atora y se acompleja si no se encuentra una respuesta rápida como veremos a continuación.

Las soluciones a este problema habitualmente suelen ser los ***NuGets,*** presentados anteriormente, **componentes personalizados** o componentes personalizados con **programación exclusiva** por cada una de las plataformas en las que programemos.

La primera de las soluciones, los *NuGets,* suelen ser una respuesta rápida y normalmente responden a las principales demandas de la comunidad; si bien es el caso, puede ser que no encontremos ninguno que se adecuen a nuestro problema o que dicho *NuGet* no cuente con una licencia libre para su uso tanto personal como a nivel comercial.

El hecho es que los *NuGets* son un recurso muy útil, sencillo y rápido y ampliamente usado en cualquier proyecto en *Xamarin*. Por ejemplo, la librería ***Acr.UserDialogs*** nos permite realizar diálogos de una forma sencilla y clara para el usuario en cualquiera de las plataformas Android, iOS o Windows Phone que de cualquier otra manera hubiese resultado compleja.

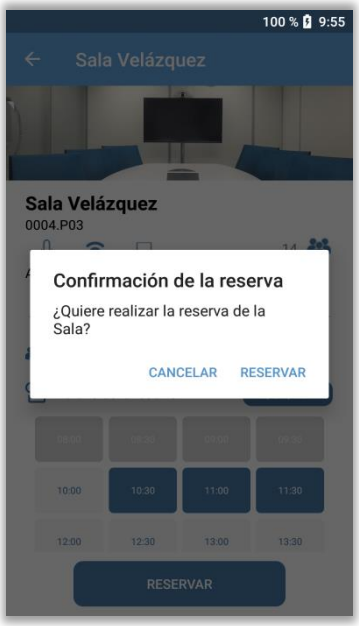


*Ilustración 20: Diálogo de tipo "Alert" generado con la librería **Acr.UserDialogs**.*

Por otra parte, los componentes personalizados son, como su nombre indica, componentes que programamos para dar una respuesta concreta a nuestro requerimiento. Dichos componentes normalmente toman como base algún componente existente en *Xamarin* y modifican su comportamiento o propiedades.

Aunque resulta una respuesta a medida, el tiempo de programación del componente no es rápida ni sencilla y requiere de un conocimiento superior comparado con el conocimiento medio para programar en *Xamarin.*

Del mismo modo, la programación de un componente personalizado siempre resulta limitada y muchas veces se ha de optar por una programación de un componente con un comportamiento exclusivo por plataforma, como veremos a continuación.

Muchas veces la limitación conlleva problemas con comportamientos y capturas de eventos, que requieren de una programación todavía más exclusiva y que, de cara al usuario, resulta trivial y necesaria. Es importante no generar expectativas en el usuario, es decir, no podemos programar un componente en pantalla que parezca que

realiza unas acciones y tienen unos controles que luego el usuario no va a poder realizar.

Volviendo a la aplicación de reserva de salas, un ejemplo de un componente personalizado lo encontramos en la pantalla principal. Las listas horizontales no existen de forma nativa en *Xamarin.Forms,* por lo que hubo de optar por programar un componente personalizado que realizase dicha función.

En este caso, no fue posible usar como base las listas verticales y por ese motivo, ciertos aspectos del componente no se llegaron a programar, por ejemplo, las animaciones al *clicar* una celda de la lista.

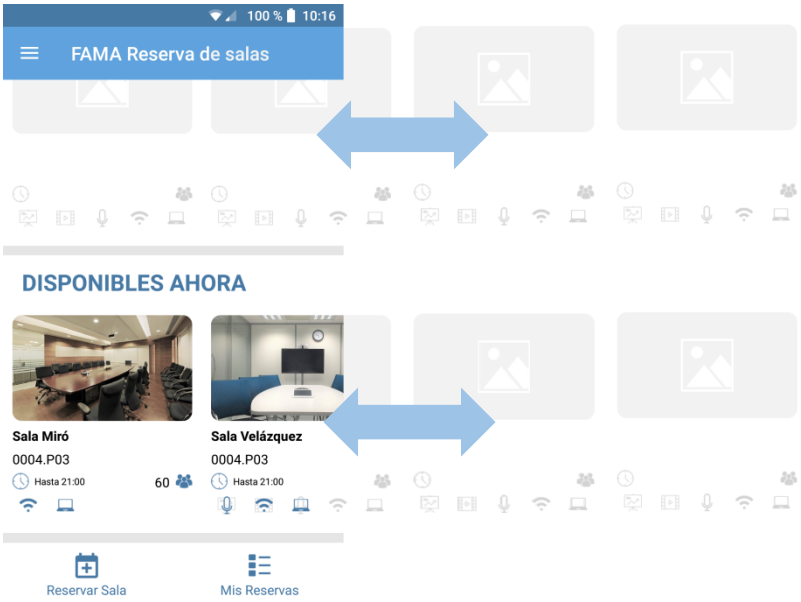


Ilustración 21: Pantalla principal de la aplicación y listas horizontales de reservas y salas.

Finalmente, los componentes personalizados con programación exclusiva por plataforma nos ofrecen el mayor control posible sobre el aspecto y comportamiento de un componente en las diferentes plataformas. Esta solución propone crear un componente dentro del proyecto común que estará ligado al mismo componente dentro de cada uno de los proyectos de cada plataforma.

A simple vista podemos ver que es el más costoso y el que requiere de un mayor conocimiento, pues no solo programaremos en C# en el proyecto *Xamarin.Forms*, si no que deberemos hacerlo paralelamente en la parte de *Xamarin.Android* y *Xamarin.iOS* respectivamente. Esto nos obliga a ser conocedores del entorno de *Xamarin.Android* y *Xamarin.iOS* y de los sistemas *Android* e *iOS* respectivamente ya que, aunque lo haremos a través de los componentes de *Xamarin,* estos basan su comportamiento en dichos sistemas y en sus mismos componentes, con sus características distintivas de cada uno.

Como ejemplo, dentro del proyecto personalizamos de esta manera el componente *picker* y el componente *slider* para poder asignarles el

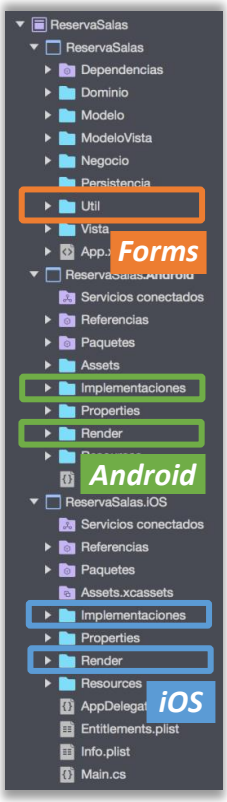


Figura 14: Estructura del proyecto Xamarin.

color que nosotros queríamos en cambio de los habituales en cada sistema por defecto.

En la estructura de carpetas, el código de estos componentes personalizados dentro de cada sistema se encuentra dentro de la carpeta *Render* e *Implementaciones* y dentro del proyecto común, en la carpeta *Útil, en Behaiviors* y *Controles* respectivamente.

Patrón MVVM

Como hemos visto anteriormente en el apartado 4.5.1, al hacer uso del patrón MVVM la vista es más independiente de la capa de datos, de tal manera que la capa de datos no se ve afectada por posibles cambios en la vista y viceversa y las partes se programan una independiente de la otra.

Dicha desconexión se suple con ciertos enlaces de los diferentes componentes a los datos y con comandos para realizar acciones pero, por otra parte, el control de la vista es limitado.

A la hora de programar ciertos componentes, como han sido el *grid* de botones de las horas de reserva o las listas horizontales, me he encontrado con verdaderas limitaciones para poder programar la lógica de datos detrás; El patrón MVVM en *Xamarin* no ofrece una respuesta completa.

De la misma forma, desde el Modelo de vista no hay manera de conocer el estado de una pantalla ni los eventos de transición tales como *OnAppearing* u *OnDisppearing* cuando la vista a la que sirven datos se muestra o desaparece respectivamente.

Esto mismo es necesario para realizar llamadas a servidor sin bloquear el proceso principal y avisar de posibles errores al usuario que de otra manera no se mostrarían sin una vista creada.

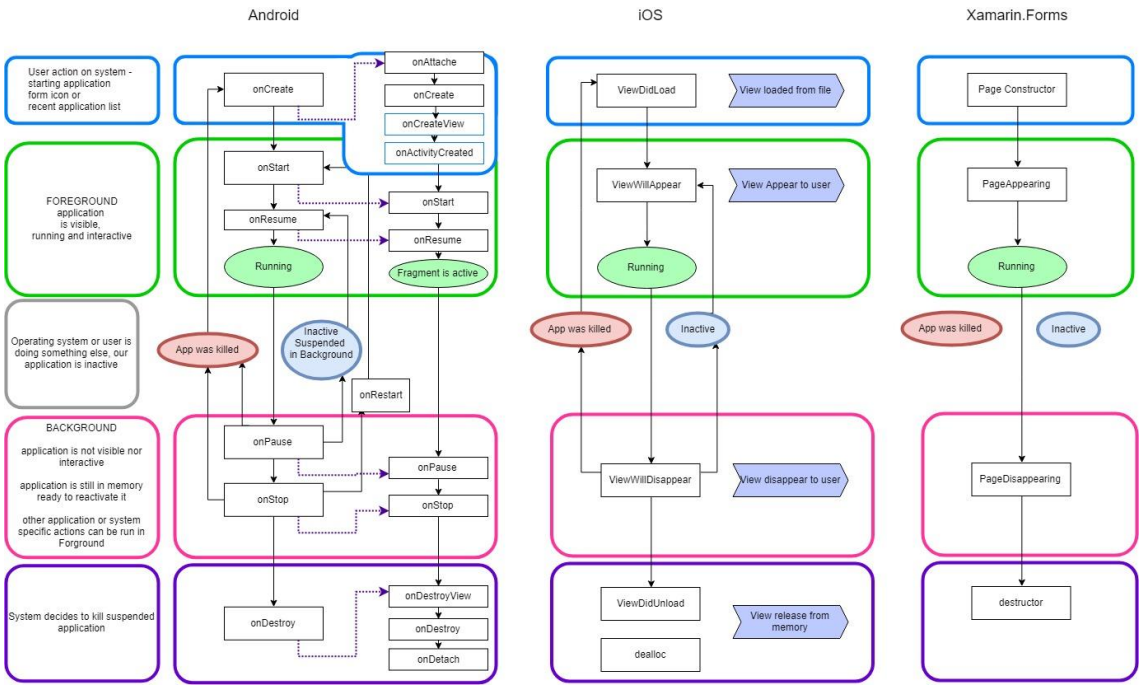


Figura 14: Comparación de los ciclos de vida de una vista en Android (izquierda), iOS (centro) y Xamarin.Forms (derecha). Fuente: https://jlamch.blogspot.com/2018/08/mobile-view-lifecycle-fixing.html.

En consecuencia, en algunas partes del proyecto se optó por complementar el patrón MVVM con una parte de código asociado a la vista para solventar dichos inconvenientes.

Del mismo modo, como la lógica del grid resultó muy compleja, se programó una *ViewContent* aparte, con su código asociado a la vista, de tal manera que se pudiese separar de la ViewPage que la contenía y del resto del código MVVM puro.

Cabe destacar que existen *NuGets* que añaden funcionalidades que complementan los recursos de *Xamarin* para un uso completo y óptimo del patrón MVVM, pero no se llegaron a usar por no ser necesario pues, con la solución expuesta anteriormente y con poco código no intrusivo dentro de la estructura general, fue suficiente.

Interfaz Android vs iOS

Uno de los temores al iniciar el proyecto, aunque ya identificado y valorado en la fase inicial, fue la visualización de las pantallas en ambos dispositivos Android e iOS.

Durante el desarrollo, y en la programación de las vistas, las pruebas fueron realizadas en ambas plataformas de forma constante y paralela por dicho motivo. De esta manera, cuando un componente no se visualizaba de forma correcta, o cuando una pantalla no se visualizaba de forma adecuada en una de las plataformas o, incluso, se identificaba un error en el diseño, este podía ser corregido de inmediato.

Aunque *Xamarin.Forms* utiliza componentes nativos en varias plataformas, es posible que dichos componentes no se vean bien en alguno de los dispositivos; normalmente, este error es debido a que muchos de los componentes se adaptan de forma distinta a la pantalla en alguno de los dispositivos, a la densidad de píxeles de las pantallas, que es interpretado de forma distinta en ambas plataformas, o que directamente, uno de los componentes tienen ajustes en sus formas nativas que no se pueden configurar directamente desde el componente de *Xamarin.Forms.*

Este inconveniente se presentó de forma notable en la pantalla de *Búsqueda de una sala,* donde los filtros con *pickers* y *sliders,* que no siguen el mismo diseño en Android e iOS, se reprogramaron nativamente con un componente personalizaron para poder modificar diferentes colores y bordes para que fueran similares.

De todos modos, uno de los puntos del diseño para dicha pantalla era poder introducir valores personalizados para las horas en ambos *pickers*, pues el diseño estaba pensado para que tan solo se pudiesen seleccionar horas cada 30 minutos, pero ni el componente lo permitía, ni con el componente personalizado se podía resolver de forma sencilla o de una forma que no requiriese una cantidad de tiempo superior a otras alternativas.

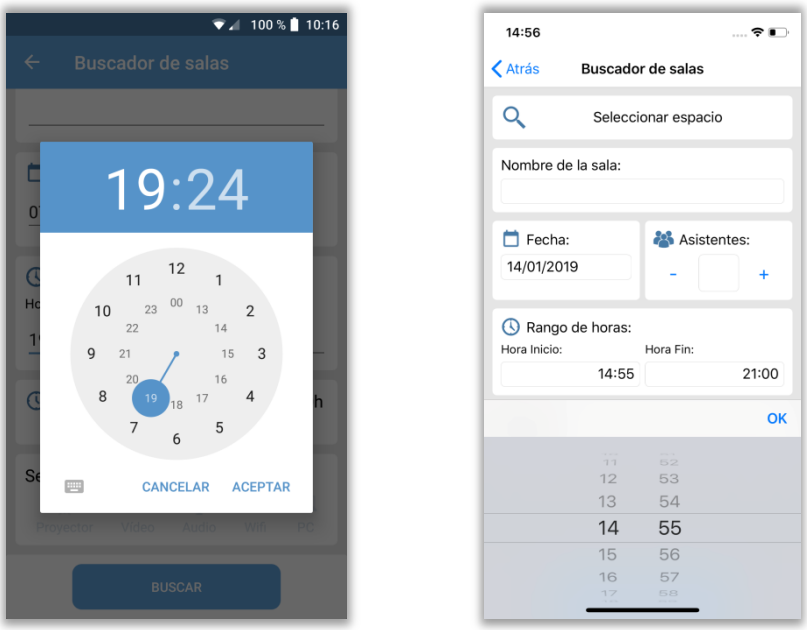


Ilustración 22: Comparación de un picker nativo Android (izquierda) y un picker nativo en iOS (derecha).

6. Plan de pruebas

Hasta esta parte, se ha hablado varias veces de la fase de pruebas de la aplicación, pero en este apartado lo desgranaremos y ampliaremos, detallando las diferentes pruebas realizadas.

Hablando del plan en general, dentro del proyecto hubo pruebas individuales por cada funcionalidad desarrollada dentro de cada iteración y dos fases de pruebas generales al finalizar cada una de las fases. Cabe destacar que las pruebas individuales de cada funcionalidad se realizaron tanto en *Android* como en *iOS* paralelamente.

Las pruebas individuales fueron sencillas pruebas de validación que tenían como objetivo confirmar la iteración y avanzar en el desarrollo. Dichas pruebas eran muy concretas y trataban la funcionalidad desarrollada en concreto y, si precedía, las funciones relacionadas con la misma.

Debido a la metodología ágil del desarrollo, las pruebas no fueron documentadas, pues se prescindió de ellas a favor de no detener el desarrollo. De todas formas explicitaremos a continuación las pruebas y los aspectos probados en cada una de las pantallas de la aplicación.

Inicio de sesión

La pantalla de *inicio de sesión* junto a la pantalla de Reserva de una sala es con diferencia de las pantallas con más validación a realizar.

Dentro de esta pantalla, la mayor cantidad de pruebas a que se realizaron guardaron relación con el inicio de sesión y la comunicación con el servidor. De forma previa, se validó que los campos introducidos por el usuario, nombre, contraseña y servidor, fueran correctos y que los posibles errores fueran capturados y mostrados de forma clara al usuario.

Dentro de estos, los errores posibles debían ser o un error en la combinación usuario-contraseña, un error en el enlace al servidor, no haber aceptado la política de privacidad o un error interno del servidor.

Como veremos más adelante en el apartado de resolución de errores, estas pruebas no fueron realizadas de forma completa por la parte de servidor y más adelante se encontraron errores.

Acerca de

La pantalla *acerca de* es una pantalla que muestra diferentes datos en relación a la sesión y el servidor y no tiene muchos elementos a validar.

Dentro de la fase de pruebas, se eliminó un botón ubicado en la parte inferior de la pantalla, que devolvía al usuario a la página anterior, por tratarse de un elemento redundante. Las fases de prueba también nos ayudaron a valorar el diseño y a realizar pequeños ajustes como el mencionado.

Menú principal

Dentro del menú principal encontramos las primeras llamadas al servidor con un número de datos significativo. Dentro de la información de las salas se encuentra las imágenes de las mismas, lo que conforma un hándicap en la comunicación.

Al principio del desarrollo de esta vista, las pruebas conformaron una parte importante para realizar diversos ajustes en las peticiones al servidor o valorar qué información mostrar en las diferentes listas.

El mayor inconveniente destacable en esta pantalla fue el número de salas a mostrar, que finalmente se decidió limitar a 10 salas por lista.

Ficha de Reserva de sala

Como hemos comentado anteriormente, la *ficha de una reserva de sala* resultó la pantalla más complicada de la aplicación, con diferentes casos de uso que llevaban al usuario a dicha pantalla, con diversas opciones y campos a con diferentes comportamientos.

Al entrar en la ficha, una gran parte de la información ya se obtiene anteriormente en la lista y la única petición a servidor consiste en la lista de reservas de la sala en el día de la consulta. Por ese mismo motivo, una gran parte de la validación fue necesario hacerlas para el *grid* de horas.

Para dicho componente se validó que las reservas descargadas se mostrasen correctamente, que no se permitiesen solapamientos y que esta componente se visualizase correctamente.

Como hemos comentado en otras partes del documento, el *grid* de horas resultó un componente que requirió la programación de una lógica que se tuvo que validar punto a punto, con todas las casuísticas posibles para la selección de franjas horarias de una reserva de sala.

Fue durante las fases de pruebas de la pantalla donde se tomó la decisión de no mostrar las filas de horarias ya pasadas, de manera que se agilizase al usuario la visualización de la disponibilidad de una sala ya que en determinadas pantallas era necesario hacer *scroll* para ver las últimas horas del día.

Por otra parte, como en otras pantallas con acciones que llevan a realizar una petición al servidor, se probaron todos los casos de error, que fueron para los casos de uso siguientes: finalización de una reserva, anulación de una reserva, modificación de una reserva y errores provenientes del servidor.

Dentro de los campos de la reserva, el número de asistentes se limitó para no exceder a la capacidad de la sala y el comentario de la misma se permitió que el usuario lo dejase en blanco, mostrando un mensaje por defecto en su creación. De este modo se agilizaba y simplificaba la reserva de una sala. Estos ajustes se hicieron para no entorpecer la experiencia del usuario y fueron ajustados durante la fase de pruebas.

Buscador de salas

La pantalla del *buscador de salas*, fue la pantalla que más cambió dentro del plan de pruebas. La mayoría de componentes se cambiaron por componentes propios de interfaces de aplicaciones tales como *sliders* y *steppers,* pues las impresiones que se recibieron de la pantalla fueron las de una pantalla tosca y que requería de muchos pasos para ser usada completamente.

Las pruebas a realizar en esta pantalla principalmente pasaron por las pruebas de los diferentes componentes y su correcto paso del valor a la siguiente pantalla. También fue probado el selector de espacios, que se hubo de rehacer, ya que el selector de la aplicación de la anterior aplicación fue descartado.

Lista Salas del buscador

Las pruebas realizadas en la *lista de salas del buscador* fueron similares a las realizadas en el menú principal: Se validó que se mostrasen bien y que se descargasen las fotografías de forma correcta además de que las salas estuviesen bien seleccionadas.

Los datos de los filtros asignados por el usuario también debían mostrarse de manera correcta, con el hándicap de que el campo de la ubicación podía contener muchos caracteres dependiendo de los niveles de espacios.

Historial

Finalmente, las pruebas del historial de reservas también fueron similares a las de la pantalla principal; las dos listas de salas se solicitaban a la vez y se debían de cargar también de forma similar.

Durante las pruebas de esta pantalla se decidió añadir a las listas un mensaje con un icono en el caso de que no hubiese elementos a mostrar para notificar al usuario.

7. Publicación

Una vez finalizado el desarrollo, hablaremos de la parte final y el broche de este proyecto; la publicación de las aplicaciones en las tiendas de aplicaciones, tanto de *Google* como de *Apple*.

Una vez creada la aplicación, lo siguiente que deberemos hacer es prepararla para su compilación para posteriormente publicarla. Esta compilación se ha de hacer en modo *Release* para posteriormente firmarla con nuestras credenciales de desarrollador.

Para la primera de las tareas *Xamarin* facilita el trabajo, pero nos encontramos con que dependiendo de la plataforma, existe todo un abanico de opciones para la compilación como veremos en los siguientes apartados.

El otro aspecto a tener en cuenta es el modo de ejecución de la aplicación durante el desarrollo. Hasta el momento se ha trabajado en modo *Debug* y hay ciertos aspectos que varían y resultan importantes.

Dentro de la aplicación existe la posibilidad de ejecutar cierto código dependiendo del modo en el que ejecutamos la aplicación.

Esta funcionalidad se usa principalmente para depurar la aplicación durante el desarrollo; Por ejemplo, en el diccionario que se encarga de la *localización* se alerta al desarrollador de una etiqueta faltante lanzando una *Excepción* si está en modo *Debug.* Por contraparte, en modo *Release* se muestra la etiqueta sin traducir, sin necesidad de detener la aplicación por dicho error.

El modo *Release* habilita optimizaciones y genera una aplicación sin ningún tipo de datos de depuración. En este tipo de compilación, gran parte del código puede eliminarse o reescribirse internamente, dando como resultado un ejecutable limpio y más rápido que el modo de depuración debido a las optimizaciones. El tamaño de la aplicación también és reducido considerablemente debido a su compresión.

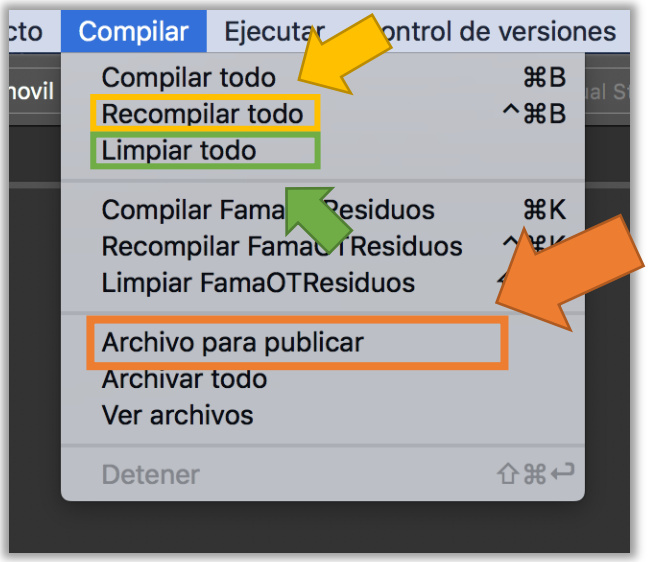


Ilustración 23: Lista de opciones del menú de compilacion en Visual Basic para Mac.

Una vez compilada la aplicación de fórma óptima para su publicación, la archivaremos para generar un código de versión para las dos aplicaciones en sus respectivos formatos: *".apk" (**A**ndroid **A**pplication Pac**k**age)* para sistemas Android y ".ipa" (**i**OS A**p**p Store P**a**ckage) para sistemas iOS.

En Android existen dos valores que nos sirven para establecer la versión de *la* app, uno es público y visible para el usuario mientras que el otro es privado y disponible únicamente para nosotros y la tienda de aplicaciones. Estos dos valores son: *android:versionCode* y *android:versionName*.

Para modificarlos es necesario acceder al archivo *AndroidManifest.xml*. En *Visual Studio* se puede hacer a través de un editor gráfico:

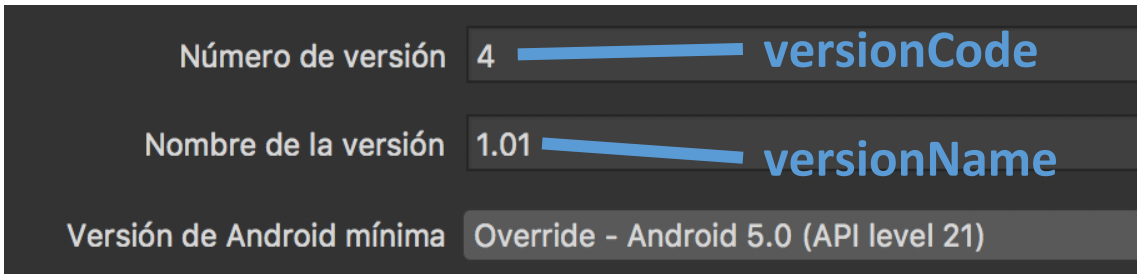


Ilustración 24: Archivo AndroidManifest.xml con los valores de Version number y Version name de la aplciación.

Para iOS la cosa no cambia mucho, igualmente existen dos valores con las mismas características (uno público y el otro privado) pero con distinto nombre: *CFBundleVersion* y *CFBundleShortVersionString* respectivamente.

Estos valores se almacenan en el archivo *Info.plist*, para el cual también se puede modificar a través del editor gráfico:

Ilustración 25: Archivo info.plist con los valores de Version y Build de la aplicación.

En este aspecto, Xamarin también nos ofrece una funcionalidad que facilita su gestión: con *Xamarin.Forms* el número de versión se unifica y el *versionCode* y el *CFBundleVersion* se usan para mostrar el número de la compilación, generando uno nuevo y normalmente aumentando de uno en uno cada vez que archivemos una compilación para su publicación.

7.1. Android y Google Play

En esta sección de tallaremos los aspectos de la compilación de la aplicación en *Android* y el proceso de subida a la tienda de aplicaciones de *Google.*

7.1.1. Compilación

De todas las opciones de compilación, en *Android* nos fijaremos en las diferentes arquitecturas de destino.

Debido al gran número de dispositivos diferentes que ejecutan un sistema *Android,* las aplicaciones deben especificar y ser compiladas sobre la arquitectura para la que estén pensadas que se ejecuten.

> *"Los diferentes dispositivos portátiles con Android incorporan diferentes CPU que, a su vez, admiten diferentes conjuntos de instrucciones. Cada combinación de CPU y conjuntos de instrucciones tiene su propia interfaz binaria de aplicación, o ABI.*
>
> *La ABI define con gran precisión la manera en que el código máquina de una aplicación debe interactuar con el sistema durante el tiempo de ejecución. Se debe especificar una ABI para cada arquitectura de CPU con la cual se desea que funcione la app."* [43]

Si bien es cierto que es posible que una *apk* contenga instrucciones para varias arquitecturas, el peso de la aplicación compilada será considerablemente mayor.

En estos casos, *Play Store* da la opción de subir varias *apk* de la misma versión de la aplicación para diferentes arquitecturas de tal manera que, de forma transparente al usuario, la aplicación esté disponible para su dispositivo si se ha subido una aplicación compilada compatible para su arquitectura.

Esta opción es determinante, pues de ella depende de forma directa el número de dispositivos que podrán consumir la aplicación y en parte, su peso.

Para este proyecto, la arquitectura objetivo fue *armeabi-v7a*, la arquitectura que recoge el mayor número de dispositivos compatibles y con mayor cuota dentro de las estadísticas de *Google,* así como el formato de dispositivo objetivo para la aplicación.

ABI	*Conjuntos de instrucciones admitidos*	*Notas*
armeabi	• ARMV5TE y posteriores • Thumb-1	Sin cálculo de punto flotante asistido por hardware.
armeabi-v7a	• armeabi • Thumb-2 • VFPv3-D16 • Otro, opcional	Incompatible con dispositivos ARMv5, v6.
arm64-v8a	• AArch-64	
x86	• x86 (IA-32) • MMX • SSE/2/3 • SSSE3	Incompatible con MOVBE o SSE4.

[43] "Administración de ABI – Google" https://developer.android.com/ndk/guides/abis

x86_64	• x86-64 • MMX • SSE/2/3 • SSSE3 • SSE4.1, 4.2	
mips	• MIPS32r1 y posteriores	Usa cálculo de punto flotante asistido por hardware y supone una relación de reloj CPU:FPU de 2:1 para obtener la máxima compatibilidad. No proporciona micromips ni MIPS16.
mips64	• MIPS64r6	

Tabla 11: Diferentes arquitecturas y conjunto de instrucciones posibles para la compilación de una aplicación Android.

7.1.2. Publicación

El proceso de subir una aplicación a la tienda de aplicaciones de google es sencillo en cuanto a pasos, pero complejo si hablamos de los detalles a realizar dentro de cada uno de ellos.

Una vez tengamos ya el archivo *.apk*, deberemos ir a la página de desarrollador de google, en el apartado de aplicaciones. Dentro de esta, tendremos que crear una nueva aplicación y especificar diferente información sobre ella.

Dentro de esta información encontraremos grosso modo, formularios de cualificación, precio y distribución, países restringidos y política de anuncios, notas de la aplicación y diferentes capturas de la aplicación para mostrar a los usuarios que accedan a la página de la aplicación.

Una vez rellenados todos los aspectos necesarios, se tendrá que decidir que versión de la aplicación subir.

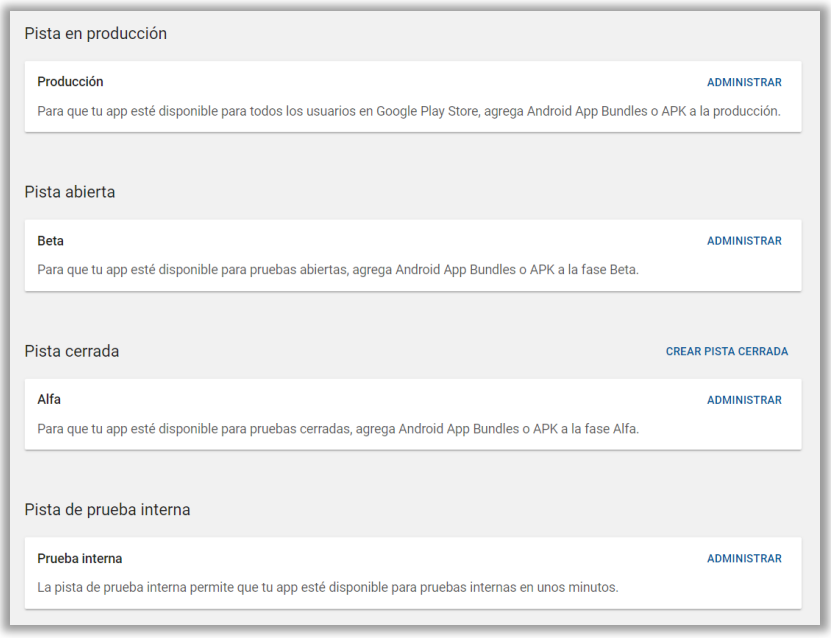


Ilustración 26: Diferentes versiones a subir de la aplicación dentro de la tienda de aplicaciones de Google.

La parte más relevante de este proceso es que, una vez subamos la aplicación en *Producción*, la *apk* pasa por un proceso de prueba y aceptación manual por parte de *Google*. Las aplicaciones en pistas cerradas y abiertas, *Alpha y Beta* correspondientemente, también pasan por un proceso de revisión pero significativamente menor.

Dichas pruebas resultan en un informe por parte de los técnicos de google con los resultados de las pruebas y los posibles errores o conflictos en el caso que llegasen a haber. Una vez aprobada, la aplicación es visible en la tienda de aplicaciones para todos los usuarios.

Con la aplicación de *Reserva de Salas,* dichas pruebas y el proceso de aceptación tardó 1 día.

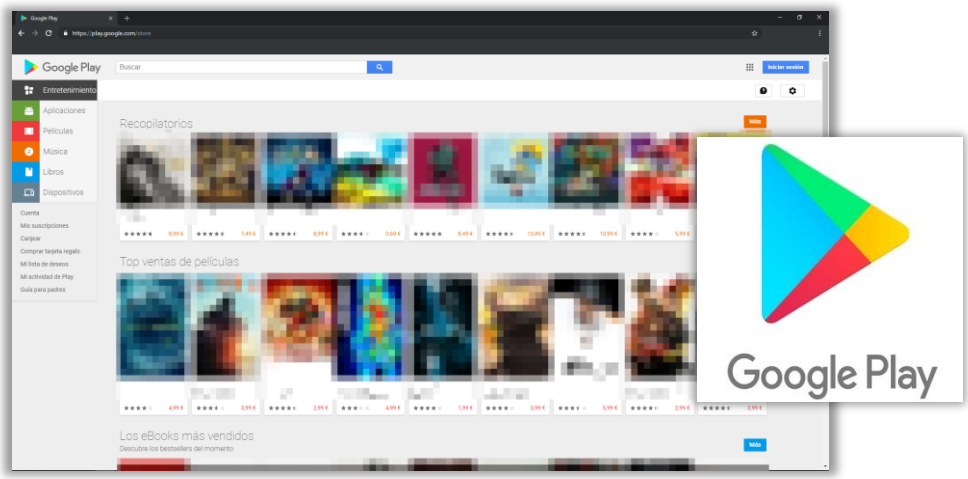


Ilustración 27: Página principal de la tienda de aplicaciones de Google y logo.

7.2. iOS y App Store

En esta sección veremos los detalles de la compilación de la aplicación para un entorno iOS y el proceso de publicación en la tienda de aplicaciones de *Apple.*

7.2.1. Compilación

Debido a que los sistemas iOS solo se ejecutan en una gama cerrada y específica de dispositivos, las opciones de compilación son menos complejas, y las diferentes arquitecturas ofrecen una mayor compatibilidad entre sí.

Al contrario de lo que ocurre en la tienda de *Google*, *Apple* obliga a subir una aplicación compilada en un sistema concreto, junto con la posibilidad de subir un compilado con diferentes ejecutables.

Arquitectura	*Dispositivo*	*Notas*
ARMv6	• iPhone (original), 3G • iPod de primera y segunda generación	

ARMv7	• iPhone 3GS, 4, 4S • iPad 1, 2, 3, Mini • iPod de tercera, cuarta y quinta generación	
ARMv7s	• iPhone 5 • iPhone 5c • iPad 4	Si el destino solo es un procesador de ARMv7s, el código generado será un poco más rápido, pero ya no se ejecutará en los sistemas ARMv7 o ARMv6 a menos que se compile un archivo binario FAT que contenga varios archivos ejecutables en el paquete.
ARM64	• iPhone 5s • iPhone SE • iPhone 6, 6 Plus • iPhone 6s, 6s Plus • iPhone 7, 7 Plus • iPhone 8, 8 Plus • iPhone X • iPad Air • iPad Air 2 • iPad Mini 2, 3, 4 • iPad Pro (todos)	Las compilaciones enviadas a App Store deben incluir compatibilidad de 64 bits; se trata de un requisito establecido por Apple. Además, iOS 11 solo admite las aplicaciones de 64 bits.

Tabla 12: Diferentes arquitecturas posibles de una compilación iOS y la lista de dispositivos compatibles.

Para este proyecto, la arquitectura de compilación objetivo fue *ARM64* ya que, con los mismos criterios que se siguieron en Android, es la arquitectura que recoge el mayor número de dispositivos compatibles y con mayor cuota activa entre los usuario.

7.2.2. Publicación

La publicación de una aplicación en *App Store* es similar al proceso descrito en la publicación de una aplicación en la *Play Store* de *Google*, aunque *Apple* solo permite dos versiones de una aplicación: dentro de un circuito de pruebas cerrado y en producción.

En este caso, al igual que con las aplicaciones *Android,* la *ipa* pasa por diferentes procesos de aprobación, tanto en la versión de pruebas internas como en producción, con sus diferencias y su exhaustividad en cada una de ellas.

Cabe decir que, dichas pruebas son más complejas en este caso que con la tienda de *Google,* por eso las pruebas internas se realizaron siempre en sistemas Android. Al cliente también se le permitió probar la aplicación durante el proceso de producción con las pruebas abiertas en *Android.*

Las pruebas, similar a lo visto en el apartado anterior, resultan en un informe por parte de los técnicos de *Apple* con los aspectos estudiados en la aplicación y, en el caso de ser rechazada, los motivos del rechazo.

El proceso de aceptación de la aplicación de *Reserva de Salas* duró 10 días desde que se envió a producción hasta que se aprobó y se lanzó en la tienda: La App fue rechazada una vez ya que los técnicos de *Apple* querían conocer el modelo de negocio de la aplicación al no existir una método para crear una cuenta para iniciar sesión en la aplicación.

Una vez se esclarecieron las diferentes preguntas sobre el modelo de negocio de la Aplicación y la manera de obtención de una cuenta, el proceso de aprobación se reanudó y se aprobó sin problemas.

Ilustración 28: Página principal de la tienda de aplicaciones de Apple y logo.

8. Futuras versiones

Una vez vistos el proceso, las principales incidencias del desarrollo, el plan de pruebas de la aplicación y su publicación, en esta sección veremos el futuro de la misma, es decir, los pasos siguientes a realizar posteriormente a este proyecto. Dentro de estos, priorizará la resolución de errores y ajustes, seguido de las mejoras y nuevas funcionalidades.

8.1. Resolución de Incidencias

La mayoría de errores detectados se han podido solucionar dentro del plazo pero, dentro de los errores conocidos que se han quedado en el tintero, comentaremos un par fácilmente identificables y su origen.

Cuando se entra por primera vez en la lista de selección de un edificio, dentro de la pantalla de *búsqueda de salas*, el dispositivo móvil detecta que el usuario tiene o no permisos para la geolocalización, pues se calcula la distancia del usuario a los mismos y, en caso de no tenerlos, se lanza la pregunta al usuario.

El problema ocurre cuando, al lanzar la pregunta de forma asíncrona, el sistema no espera a la respuesta del usuario, interpretando una localización nula y, sin posición, se captura el error y no se lanza la petición a servidor, dando como resultado una lista de espacios vacía tanto si el usuario permite al dispositivo acceder a la localización o no. Este error solo ocurre en dispositivos *Android,* no afectando así en sistemas *iOS*.

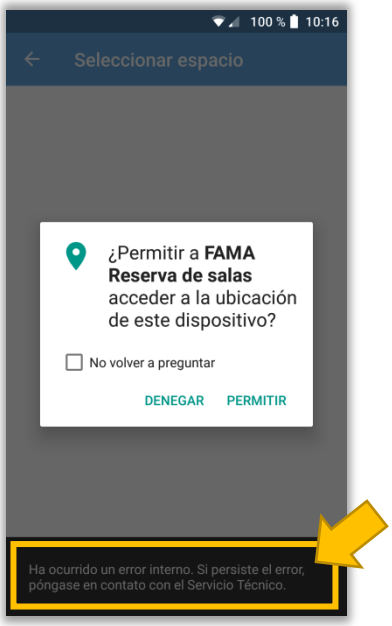


Ilustración 29: Reproducción del error de permisos de ubicación en la vista de Selección de espacios.

Otro de los errores conocidos lo podemos encontrar en el *login*. Este error viene heredado de la aplicación anterior, pues comparten el método de inicio de sesión y fue detectado durante una actualización de servidor.

El error ocurre en la interpretación de errores dentro de la llamada a servidor; por cómo se programó el servidor, cuando ocurren un tipo de errores concretos, el servidor no los captura dentro de la respuesta y la aplicación no los interpreta, dando

como resultado, un cuadro de aviso al usuario con un mensaje de error vacío, sin dar ninguna indicación al usuario más que el alarmante título de “Error”.

8.2. Mejoras

Como se ha comentado anteriormente, después de la reunión que se realizó al finalizar la primera fase, a través de las dos tiendas de aplicaciones se distribuyó una versión beta para que la aplicación pudiese ser probada. De esta manera, ya de forma temprana se tuvo un feedback directo de toda la gente que había usado la aplicación tanto a nivel funcional como a nivel estético.

De este feedback, comentaremos algunos de los cambios sugeridos y que se plantean implementar en futuras versiones:

Una vez el usuario ha iniciado sesión y ha pasado la pantalla de *login*, el usuario tiene acceso a todas las funcionalidades de la app sin restricción y sin la necesidad de iniciar sesión. De la misma forma, si la aplicación queda en segundo plano, ya sea por la ejecución de otra aplicación o por que el usuario navega al menú principal sin cerrarla, al reanudarse el usuario volverá a la pantalla donde se quedó la aplicación antes de perder el foco.

En cambio, si la aplicación se cierra, la aplicación volverá a mostrar la pantalla de inicio de sesión, pidiendo al usuario que inicie sesión de nuevo, aunque los datos de inicio de sesión estarán guardados.

Respecto a este comportamiento, visto el feedback recibido, en futuras versiones se cambiará la manera de iniciar sesión de manera que la aplicación pueda guardar la sesión una vez se haya accedido a la aplicación y esta acceda directamente al menú principal en vez de requerir el *login* de nuevo.

Otras mejoras propuestas las agrupa el buscador de salas; debido a la cantidad de campos a rellenar, los usuarios han visto que dichos campos se han de rellenar muchas veces: En primer lugar, se ha propuesto que, el espacio por defecto mostrado pueda ser cambiado, ya sea mediante una ventana de ajustes o, por defecto recuerde el último espacio por el usuario. Se deberá estudiar cual caso beneficiaría más al usuario y le ofrecería un comportamiento más natural e intuitivo.

Por otra parte, al realizar una búsqueda y acceder a una reserva mediante la lista resultante de la búsqueda, se ha propuesto que algunos de los datos de la búsqueda ya aparezcan reflejados en la pantalla de reserva, de manera que el usuario deba de introducir los parámetros de la reserva una vez.

Como conclusión, hace falta comentar que, las fuentes de *feedback* no solo son internas, pues las secciones de comentarios de las diferentes tiendas donde se ha publicado la aplicación así como los clientes directos de la misma son las fuentes más importantes y directas de opiniones.

Estas opiniones han de ser estudiadas rigurosamente, ya que el usuario final será el que llegue a valorar si la aplicación cumple con lo prometido o lo esperado, es accesible e intuitiva y, en general, si cumple con todos los puntos de los cuales partía el desarrollo.

8.3. Nuevas funcionalidades

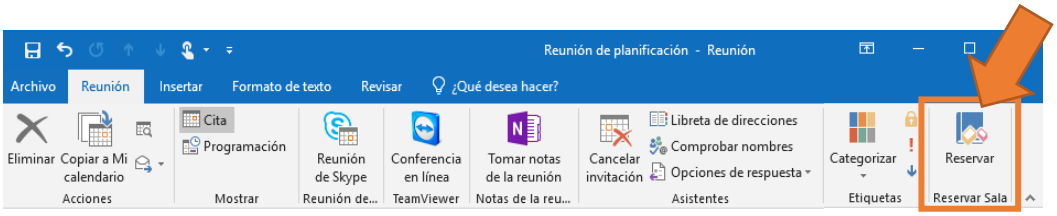
La aplicación de reservas ha resultado concisa y clara en cuanto a su funcionalidad aunque, de forma natural, se ha pensado en una ampliación de la funcionalidad de reservar sala con otra funcionalidad paralela muy usada junto a la reserva de una sala.

Esta ampliación propone dos cosas: por una parte, al realizar una reserva de sala desde la aplicación, el usuario tendrá la posibilidad de indicar diversos correos de personas o incluso directamente contactos de la agenda de manera que, si se trata de una reunión, una sala de trabajo, etc… los asistentes reciban un correo notificando las características de la reserva como el nombre de la sala, el nombre del edificio o la hora de reserva.

Por otra parte, se propone dar un paso más y, al indicar los diferentes correos y contactos, poder enviar junto al correo el evento de la reserva, de manera que todas las personas notificadas puedan añadir la reserva a su calendario. Esta función se pensó pensando directamente con la idea de integrarla con el cliente de correo de *Microsoft*, *Outlook*.

De esta última idea, se ha pensado en dar otro paso más y ampliarla de otro modo; las reservas se puede realizar desde la aplicación web FAMA/AFM y ahora desde una aplicación móvil, pero los usuarios que realizan dichas reservas muchas veces realizan el camino contrario.

Aunque ya no hablamos de un entorno de la propia empresa, una buena manera de completar el desarrollo de este proyecto resultaría en la confección de un *plungin* para *Outlook* que permitiese realizar una reserva, de una forma rápida, desde el propio cliente, y de esta manera complementar la programación de una reunión sin necesidad de salir del entorno habitual de uso del usuario.



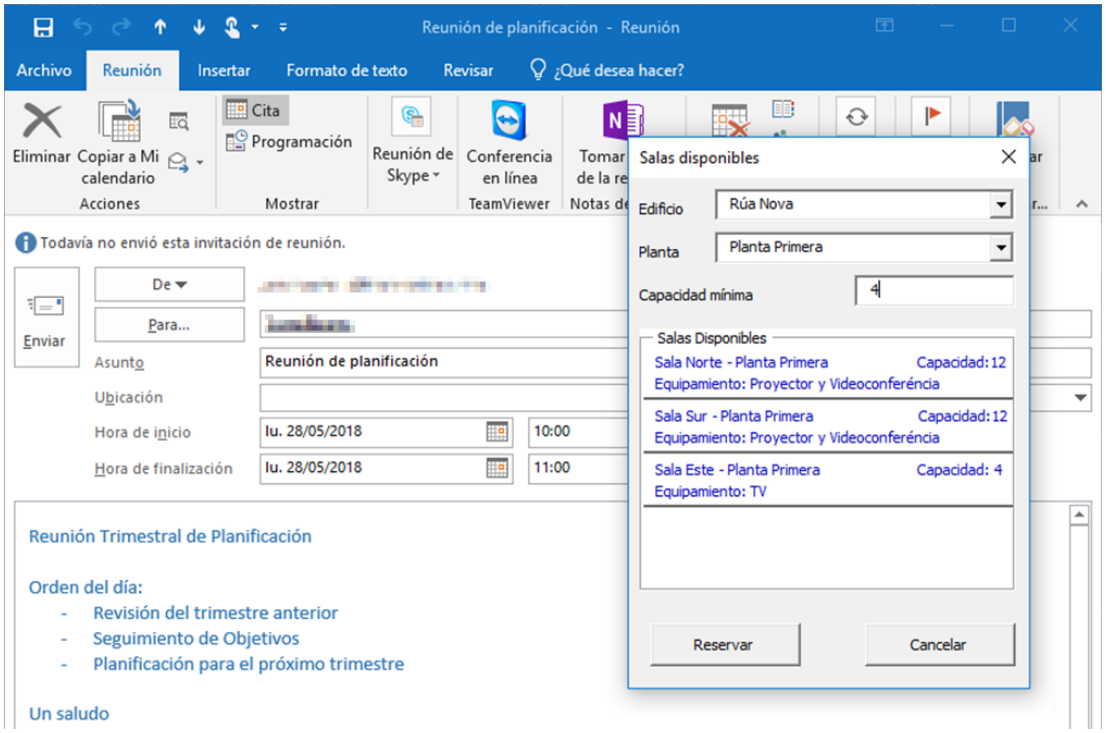


Ilustración 30: Prototipo del plungin de reserva de salas para el cliente de correo Outlook.

9. Aspecto final

En este último apartado del documento veremos finalmente el aspecto de la aplicación, por fuera y por dentro, con las diferentes pantallas de la aplicación y el mapa de clases interno respectivamente.

9.1. Pantallas

Las capturas de pantalla fueron tomadas con los siguientes dispositivos:

Dispositivo	*Características*
iPhone X	• **Sistema operativo**: *iOS 12* • **Pantalla**: *5.8", 2436 x 1125 pixels*
Mi A1	• **Sistema operativo**: *Android 9 Pie* • **Pantalla**: *5.5", 1920 x 1080 pixels*

9.1.1. Inicio de sesión

La pantalla de inicio de sesión es una vista sencilla y tradicional en cuanto a pantallas de *login* se refiere. Desde el diseño inicial se tuvo claro que también debía servir como puerta de acceso a la web de la empresa y a la política de privacidad, que se aceptaría de manera obligatoria para poder iniciar sesión.

Desde esta pantalla también se puede acceder a la pantalla *Acerca de* para poder consultar diversa información sobre la aplicación.

Ilustración 31: Pantalla de Inicio de sesión. Android (izquierda), iOS (derecha).

9.1.2. Menú principal

El menú principal es la primera vista que encuentra el usuario al iniciar sesión. Desde su diseño se buscó que esta pantalla ofreciese la mayor información que el usuario pudiese necesitar la mayor parte del tiempo.

En esta pantalla el usuario podrá ver dos listas principalmente, la lista de sus reservas y salas disponibles para reservar en ese momento. La información mostrada sobre las reservas y salas es la siguiente: Nombre de la sala, Edificio, capacidad, servicios de la sala y Horas inicio y fin de la para las *Reservas* u hora fin de la posible reserva para *Salas disponibles*.

Clicando sobre los ítems de la lista superior, se puede acceder a la vista de *Reserva de Sala*, donde se mostrará la información sobre una reserva que hayamos realizado. Por otra parte, si se clica sobre un ítem de la segunda lista, accederemos a una pantalla similar pero con la información de la disponibilidad de una sala para ese día y con un una disponibilidad en ese momento.

En una de las iteraciones del desarrollo, como se ha comentado en el apartado de desarrollo, se eliminó otra lista de salas disponibles por sobrecargar la pantalla y confundir más que ayudar al usuario.

Debajo de la pantalla y fijados al final de esta, encontramos los botones de *Reservar Sala* y *Mis Reservas,* que nos llevarán a la vista del *Buscador* y al *Histórico* respectivamente.

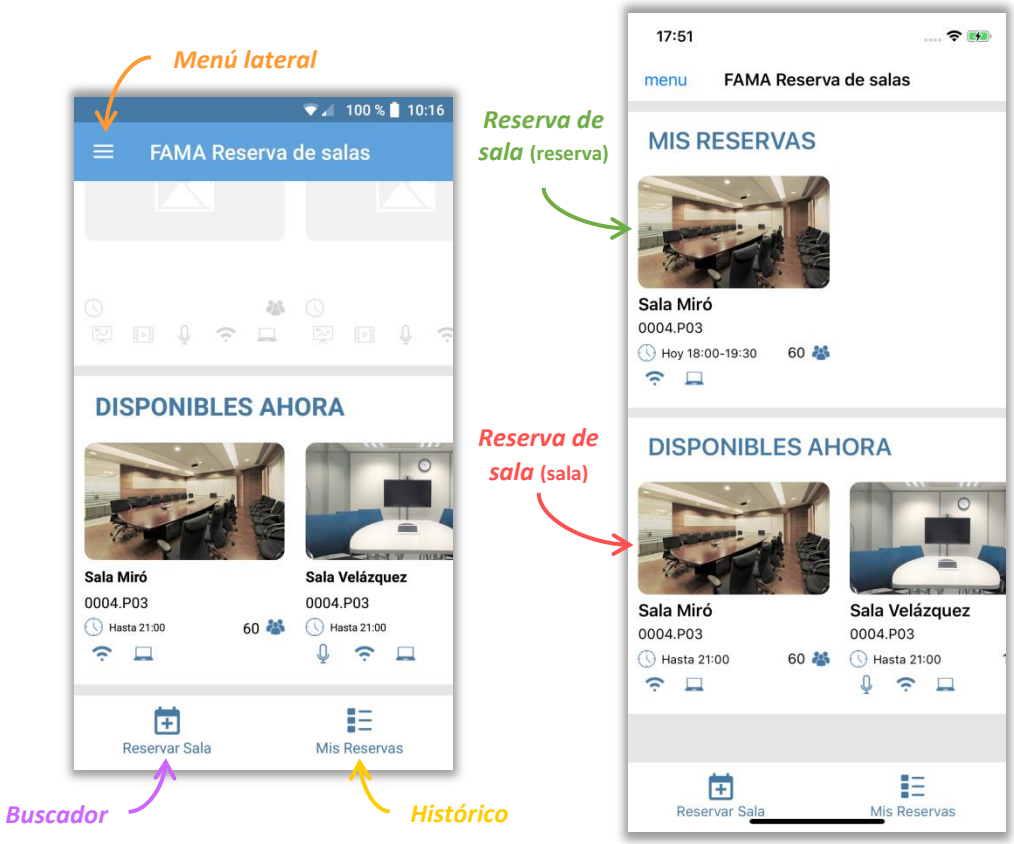


Ilustración 32: Pantalla del menú principal. Android (izquierda), iOS (derecha).

En esta pantalla también disponemos de un menú lateral que nos da acceso a una web de contacto de la empresa, la pantalla *Acerca de*, que ahora contendrá información de la sesión, y la opción de *cerrar sesión* que nos devolverá a la pantalla de *inicio de sesión.*

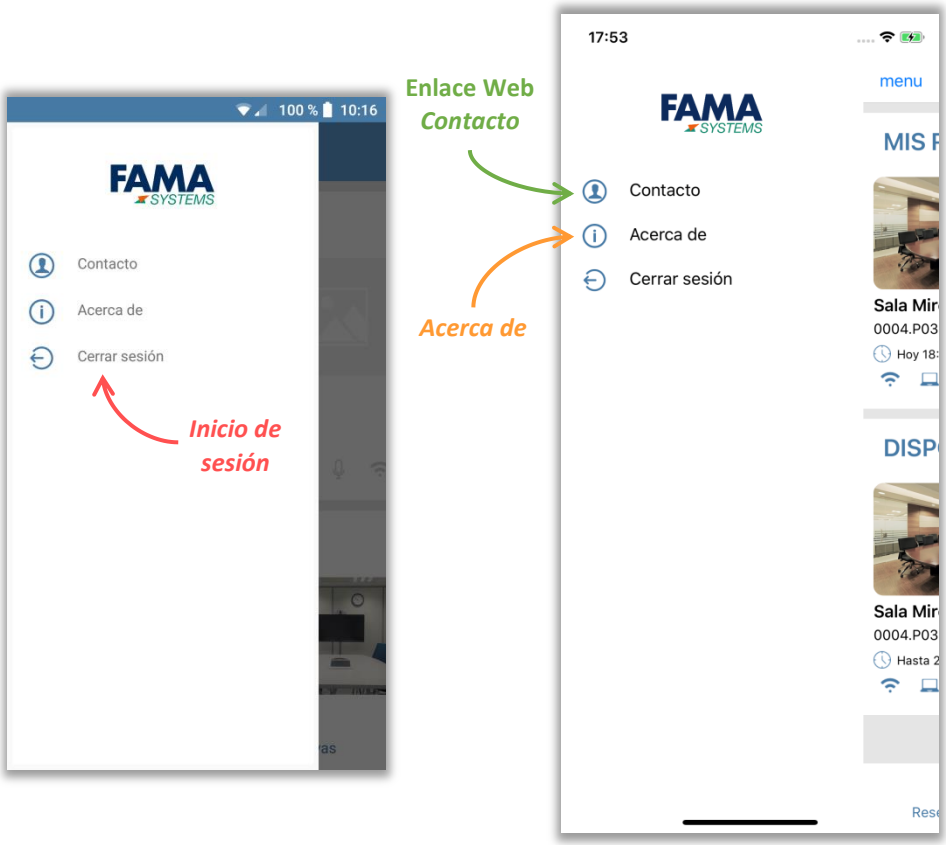


Ilustración 33: Pantalla del menú principal con el menú lateral abierto. Android (izquierda), iOS (derecha).

9.1.3. Acerca de

En la pantalla *Acerca de,* el usuario podrá ver una pequeña descripción de la aplicación, el servidor en el cual haya iniciado sesión, su usuario, el idioma de la aplicación y la versión. También podrá acceder a la web con la política de privacidad de la empresa clicando sobre el enlace en el inferior de la pantalla.

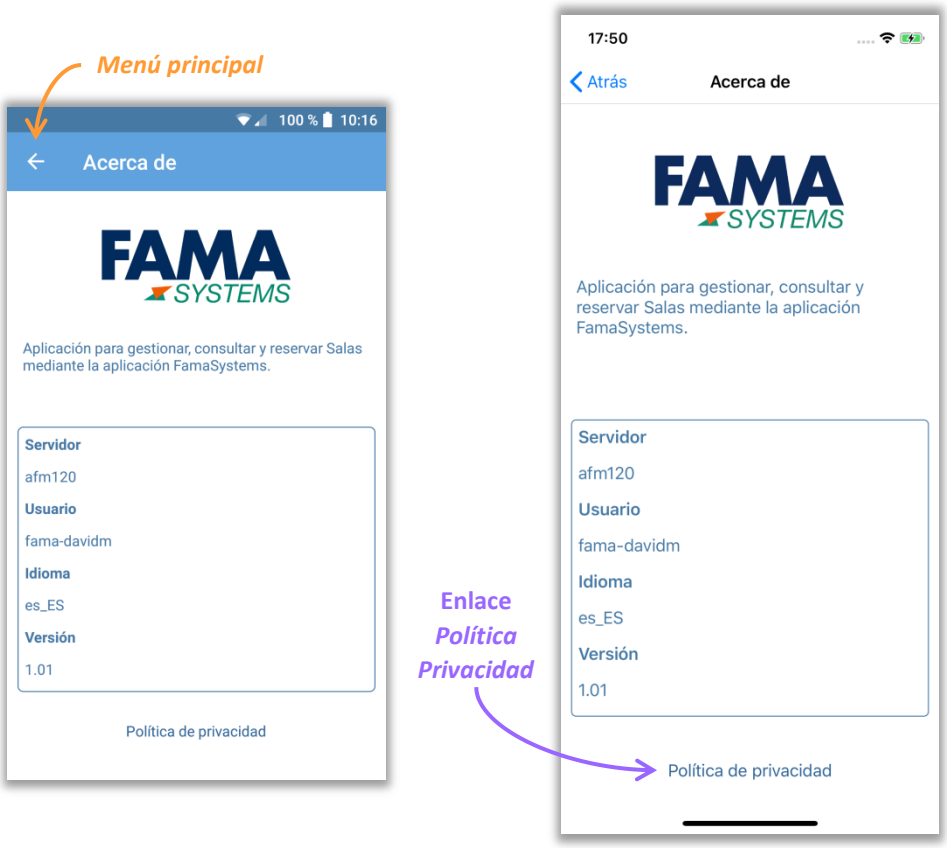


Ilustración 34: Pantalla Acerca de. Android (izquierda), iOS (derecha).

9.1.4. Reserva de Sala

Dentro de esta aplicación, la pantalla de *Reserva de Sala* es sin duda la pantalla más importante y en la que se centran todas las vistas. De una manera u otra, todas las vistas llevan al usuario a una *Reserva propia* o a una *Sala disponible.*

En esta vista se muestran las propiedades de una reserva, informadas o no dependiendo de si se trata de una *Reserva* o una *Sala.* La parte más importante es el *grid* de horas, donde el usuario podrá seleccionar las horas que desee para reservar siempre que estén disponibles.

En ambos casos, las horas que ya hayan pasado estarán marcadas con un gris claro y las horas ocupadas por otras reservas se mostrarán en gris oscuro (Ilustración 35 e Ilustración 36). Dentro de una *Reserva de sala,* las horas de la reserva se mostrarán en verde oscuro (Ilustración 36).

En la parte inferior se encuentran anclados los botones con las diferentes acciones que pueden realizar los usuarios sobre *Reservas de sala* y *Salas disponibles*. Dentro de una *Sala disponible* el botón que se mostrará será el de *Reservar* mientras que en una *Reserva de sala* se mostrarán el botón de *Modificar* y los botones de *Finalizar* o *Anular* dependiendo de si la Reserva es en ese mismo instante o futura (Ilustración 36).

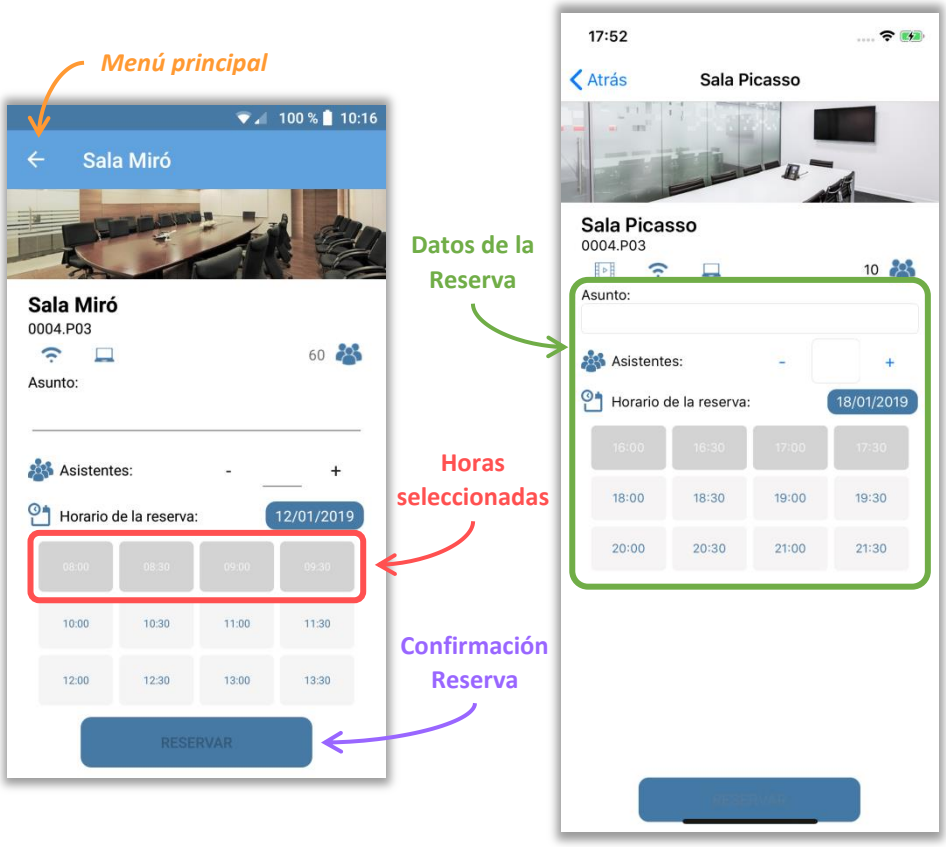


Ilustración 35: Pantalla de Reserva de Sala. Android (izquierda), iOS (derecha).

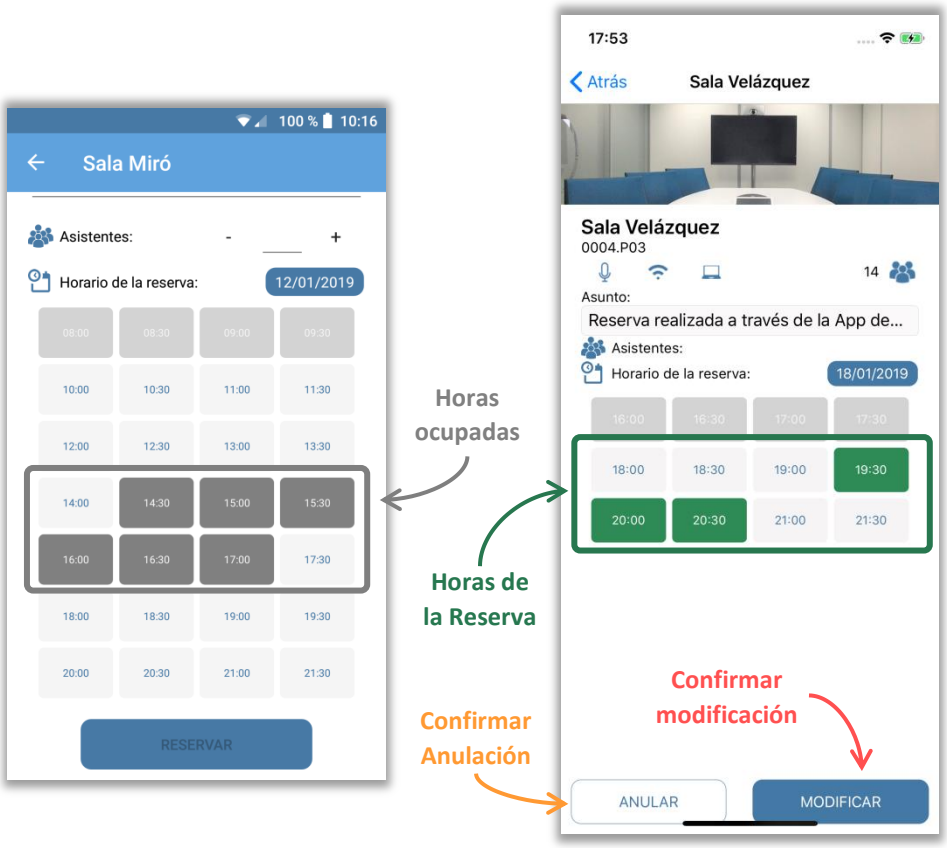


Ilustración 36: Pantalla de Reserva de Sala con salas ya reservadas. Android (izquierda), iOS (derecha).

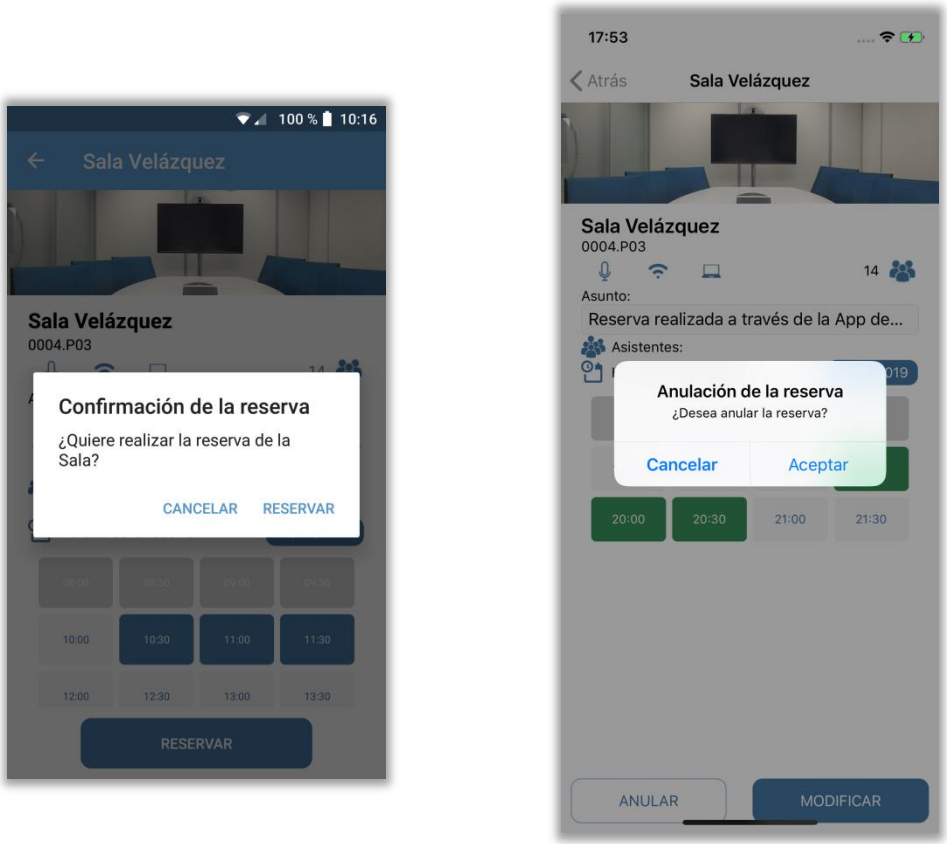


Ilustración 37: Pantalla de Reserva de Sala. Detalles de los mensajes de confirmación para el usuario. Android (izquierda), iOS (derecha).

9.1.5. Buscador

Después de la pantalla de *Reserva de Sala,* la pantalla del *Buscador* es la pantalla más compleja en cuanto a componentes se refiere.

En esta pantalla encontramos todos los aspectos de una Reserva de sala de forma parametrizable, de tal manera que conformen los filtros en la búsqueda de una sala que cumpla dichas características al pulsar sobre el botón de *Buscar*.

En la parte superior podemos encontrar un botón de búsqueda que nos llevará a una pantalla con la lista de todos los espacios y sus plantas que contienen salas que permitan reservas.

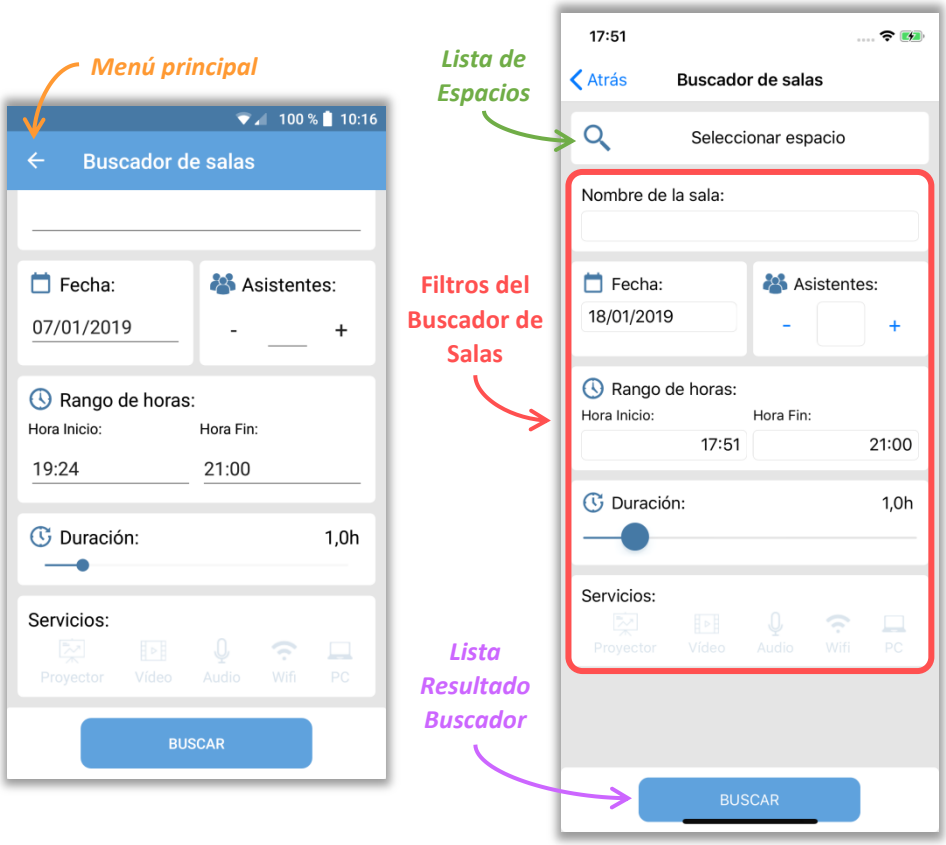


Ilustración 38: Pantalla de Buscador de Salas. Android (izquierda), iOS (derecha).

Lista de espacios

Dentro de la pantalla *Lista de espacios* podremos encontrar la lista de espacios que contienen salas para reservar ordenadas de forma ascendiente por cercanía al usuario.

A demás, estos espacios estarán organizados por niveles de tal manera que el usuario pueda seleccionar una planta o un espacio concreto de un edificio u espacio.

Para seleccionar un espacio entero, el usuario deberá pulsar el botón de *Ok* en la parte superior izquierda dentro del espacio elegido.

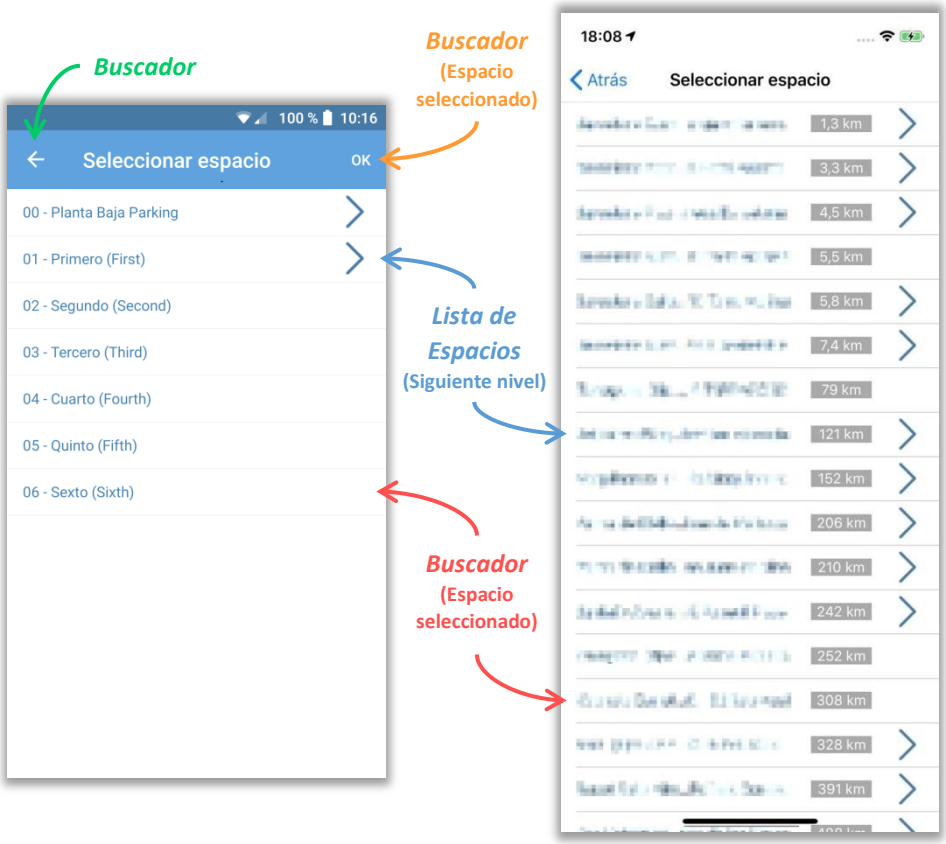


Ilustración 39: Pantalla de la Lista de espacios dentro del buscador de salas. Android (izquierda), iOS (derecha).

9.1.6. Lista resultado Buscador

En esta pantalla, el usuario podrá ver una lista de salas disponibles similar a la del menú principal pero de forma vertical.

Las salas mostradas en esta lista corresponderán las características especificadas por el usuario en la vista de *Buscador*, que se muestran en la parte superior de la pantalla.

Al clicar sobre una de las salas, se accederá a la pantalla de *Reserva de Sala* para que el usuario pueda realizar una reserva.

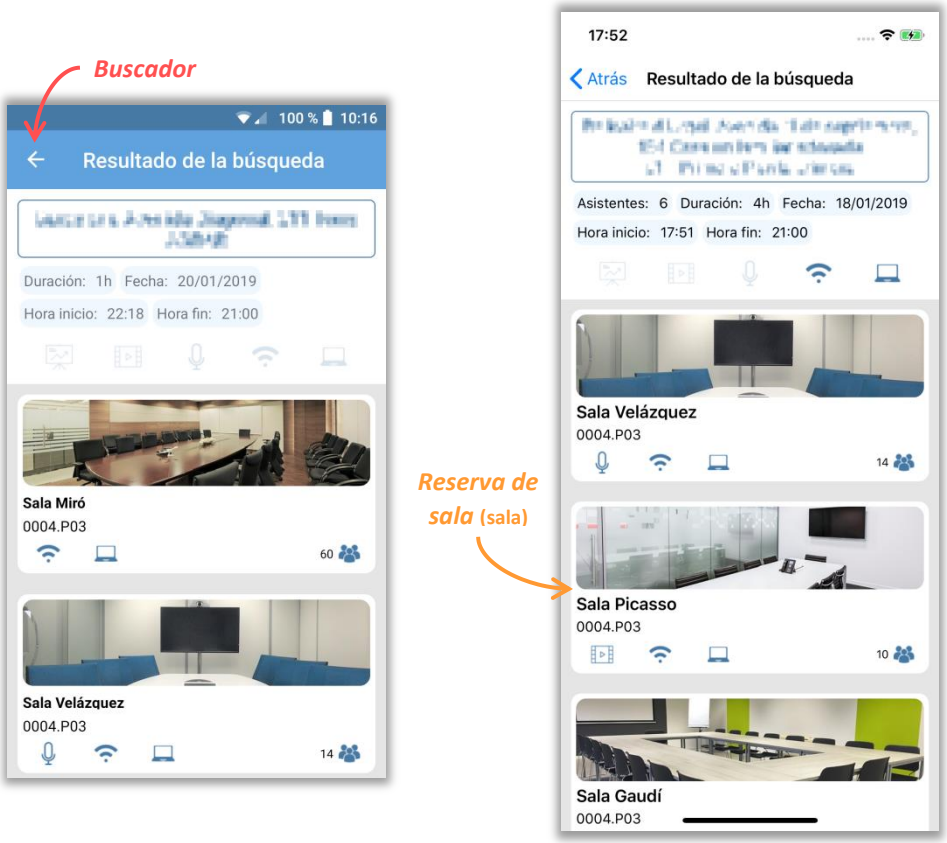


Ilustración 40: Pantalla de Lista de salas del buscador. Android (izquierda), iOS (derecha).

9.1.7. Histórico

Esta pantalla es una *TabbedPage, donde se muestran las reservas del Usuario en la misma página en diferentes pestañas.*

Reservas futuras

La pestaña de *Reservas Futuras* muestra todas las reservas del usuario futuras y en proceso en ese mismo instante.

Clicando en una Reserva, el usuario podrá ver todos los detalles de la reserva, así como también tendrá las opciones de Editar, Anular o Finalizar según convenga.

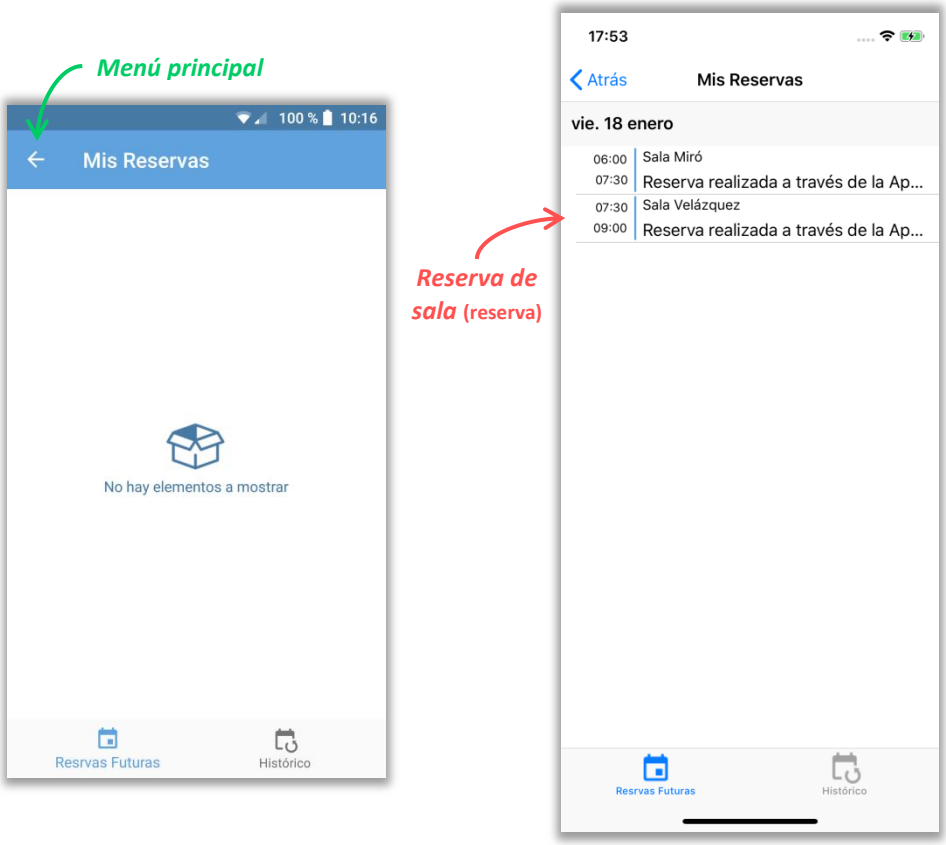


Ilustración 41: Pantalla de Reservas futuras dentro de la pantalla de mis reservas. Android (izquierda), iOS (derecha).

Histórico

Por contraparte, en la pestaña de *Histórico* el usuario podrá encontrar todas las *Reservas de Salas* anteriores al momento actual, con un límite de 365 días.

Si el usuario clica sobre alguna Reserva, podrá acceder a la pantalla de *Reserva de Sala,* donde podrá consultar las horas de la Reserva y los datos en modo de consulta. Evidentemente, no estarán disponibles ninguna de las funciones de una *Reserva de sala* Actual o futura.

Ilustración 42: Pantalla del Histórico dentro de la pantalla de mis reservas. Android (izquierda), iOS (derecha).

9.2. Mapa de clases

A continuación se mostrará el mapa de clases final de la aplicación, con los tres grandes proyectos: *Xamarin.Forms*, *Xamarin.Droid* y *Xamarin.iOS.*

ReservaSalas

Vista	Modelo de Vista	Modelo	Dominio	Negocio	Persistencia	Útil
AcercaDeVista★	*AcercaDe-ModeloVista*	*EspacioModelo*	*Espacio*	*ServicioWeb-Sesion*	*Ajustes*	*Encriptador*
LoginVista★	*LoginModeloVista*	*SalaReservaModelo*	*Filtro*	*ServicioWeb-Documento*		**Útil - Controles**
BuscadorSalas★	*BuscadorSalas-ModeloVista*	*ServicioSalaModelo*	*ItemMenuLateral*	*ServicioWeb-Espacio*		*CustomPicker*
BuscadorSalasLista★	*BuscadorSalasLista-ModeloVista*	*SesionModelo*	*RespuestaWS*	*ServicioWeb-SalaReserva*		***CustomEntry***
PaginaPrincipal★	*PaginaPrincipal-ModeloVista*	*SesionPosicion*	*SalaReserva*			*Animation-Extension*
ReservasHistorico★	*ReservasHistorico-ModeloVista*		*ServicioSala*			*HoritzontalList*
PaginaPrincipalBase	*SalaReserva-ModeloVista*					*Horitzontal-DynamicList*
SalaReserva★						**Útil - Converters**
						InverseBool-Converter
						IsNullConverter
						MostrarDistancia-Converter
						SliderToHoursString-Converter
						Útil - Diccionario
						Diccionario
						Diccionario.ca
						Diccionario.en
						PlatformCulture
						Traductor

★ Vistas con lógica asociada

ReservaSalas.Droid

Render	Implementaciones
CustomPicker	***Localize***
CustomEntry	

ReservaSalas.iOS

Render	Implementaciones
CustomPicker	***Localize***
CustomEntry	

Vista - Elementos	ModeloVista - Elementos
ListaEspacios	*BaseModeloVista*
CeldaEspacio	*MenuLateral-ModeloVista*
CeldaBuscador	
CeldaHistorial	
CeldaSalaPrincipal	
ElementoFiltro	
PaginaVacia	
GridDeBotones	
MenuLateral★	

★ Vistas con lógica asociada

Tabla 13: Mapa de clases del proyecto común de Reserva de Salas.

10. Conclusiones

El desarrollo de aplicaciones para móviles es un mundo en constante crecimiento que se ha vuelto una parte inexorable del como entendemos a día de hoy los dispositivos móviles y como interactuamos con ellos.

Más en concreto, dentro de cualquier empresa que se dedique a ofrecer un servicio de *facility magement,* es una poderosa herramienta que como hemos visto al inicio de este proyecto, ofrece una manera muy fácil de conectar al cliente con el servicio ofrecido.

Aunque el software de la empresa en la que he realizado este proyecto es grande y cubre las necesidades del cliente, opino que integrar diferentes aplicaciones a este entorno es el paso natural siguiente dentro de este mundo y da un valor añadido al producto y a la empresa muy significativo.

En cuanto a las prácticas, dentro del periodo que estuve trabajando con FAMA antes de empezar este proyecto, pude trabajar con un programa muy grande, bien definido y complejo, lo que me ha ofrecido un punto de vista que de otro modo no habría tenido a nivel profesional. En este periodo pude ver cómo funcionaba, facilitándome el trabajo de cara al desarrollo de la parte web del proyecto.

Una vez dentro del proyecto, mi experiencia ha sido totalmente diferente: Aunque ya existía un proyecto anterior, el reto contaba con muchos frentes y, tener como punto de partida una aplicación anterior poco madura dentro del desarrollo de *Xamarin* no siempre resultó beneficioso.

Por otra parte, aunque *C#* no ha sido un lenguaje al que me haya costado adaptarme viniendo de *Java*, tuve que adecuarme a sus métodos y características. Lo mismo ha ocurrido con el entorno de *Xamarin Forms* de forma más notable*; Xamarin es un* un *framework* que cuenta con sus herramientas y su propio flujo y rutinas, por lo que he tenido que aprender (y no he dejado de aprender durante el proyecto) a programar con él.

Cabe decir que al haber trabajado con *Android* anteriormente, muchos conceptos me han sido familiares, aunque *Xamarin* cuenta con su propio comportamiento y sus flujos de ejecución debido a la traducción en los entornos de *Android, iOS y Windows Phone.*

Del mismo modo, en el proyecto he jugado muchos papeles distintos; desde *back-end* puro en la parte servidor, al *front-end* más completo con el rediseño de pantallas para aumentar la accesibilidad y hacer más sencilla y útil la aplicación en general.

Realmente haciendo este proyecto siento que he puesto en práctica mucho del conocimiento adquirido en muchas de las asignaturas de la carrera, lo que he podido ver reflejado en este documento con toda la biografía consultada, apuntes y material de las asignaturas cursadas en el Grado.

Ya para finalizar, me llevo un sabor agridulce con *Xamarin* y más en concreto con *Xamarin Forms.*

Aunque sin duda hablamos de un potente kit de herramientas para desarrollar aplicaciones *multiplataforma* de forma simultánea, efectiva y útil, en ciertos aspectos me ha dado la sensación de ser un *framework* por madurar, con ciertas carencias que en algunos momentos del desarrollo se han convertido en impedimentos notables que han obligado a cambiar de rumbo o a dedicarle más horas de las esperadas.

11. Biografía

"El 'facility manager', el gran desconocido que puede ahorrar millones a una empresa - El Confidencial." https://www.elconfidencial.com/vivienda/2014-03-14/el-facility-manager-el-gran-desconocido-que-puede-ahorrar-millones-a-una-empresa_96490/

"Facility management - Wikipedia, la enciclopedia libre." https://es.wikipedia.org/wiki/Facility_management

"Qué es IFMA – IFMA España." http://ifma-spain.org/que-es-ifma/

"What is REST – Code Academy." https://www.codecademy.com/articles/what-is-rest

"Transferencia de Estado Representacional - Wikipedia, la enciclopedia libre." https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional

"GUÍA PARA CITAR SIN COMETER PLAGIO - Manual del alumno1 de la Facultad de Economía y Empresa. Curso 2013-14." https://www.um.es/documents/378246/2093234/GU%C3%8DA+PARA+CITAR+SIN+COMETER+PLAGIO.pdf/e10e72ec-faae-4a41-9520-b186dc6096a5

"¿Cómo citar y referenciar páginas web con normas APA? – Normas APA." http://normasapa.com/como-citar-referenciar-paginas-web-con-normas-apa/comment-page-21/

"SOAP - Wikipedia, la enciclopedia libre." https://ca.wikipedia.org/wiki/SOAP

"Cross-platform Frameworks for Mobile Development – Medium." https://medium.com/@MasterOfCodeGlobal/best-10-android-frameworks-for-building-android-apps-d2d0ee48e464

"The Good and The Bad of Xamarin Mobile Development – AltexSoft, software r&d engineering." https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/

"Gestión de proyectos Guía docente - GEP – Gestió de Projectes" (Acceso restringido) https://atenea.upc.edu/pluginfile.php/2527035/mod_resource/content/3/Temario.pdf

"Intuitive and Beautiful Project Planning – TeamGant" https://www.teamgantt.com/

"Presentación APP Reservas Salas (2018) – Documentación interna FAMA"

"Diseño funcional APP de Salas (2018) – Documentación interna FAMA"

"Diseño funcional APP de Incidencias (2018) – Documentación interna FAMA"

"Diseño funcional APP de Solicitudes (2018) – Documentación interna FAMA"

"Analisis de Requisitos APP de Solicitudes (2018) – Documentación interna FAMA"

"Plan de proyecto APP Solicitudes (2018) – Documentación interna FAMA"

"Planificación APP Solicitudes (2018) – Documentación interna FAMA"

"Descriptivo funcional DAMA AFM v11 (2017) – Documentación interna FAMA"

"Costes directos e indirectos de un proyecto – OBS Business School" https://www.obs-edu.com/es/blog-project-management/viabilidad-de-un-proyecto/costos-directos-e-indirectos-de-un-proyecto

"Calcular amortización de un producto" https://tutobasico.com/calcular-amortizacion/

"Activos, amortización y depreciación" https://debitoor.es/guia-pequenas-empresas/contabilidad/activos-amortizacion-y-depreciacion

" La vida útil real: cómo determinar cuando la renovación de un dispositivo de Apple pasa de ser capricho a necesidad" https://www.applesfera.com/iphone/la-vida-util-real-como-determinar-cuando-la-renovacion-de-un-dispositivo-de-apple-pasa-de-ser-capricho-a-necesidad

"Versiones de Office y conjuntos de requisitos – Microsoft" https://docs.microsoft.com/es-es/office/dev/add-ins/develop/office-versions-and-requirement-sets

"Licenciamiento de PL/SQL Developer – ComponentSource" https://www.componentsource.com/es/product/pl-sql-developer/licensing

"BOE" https://www.boe.es/boe/dias/2009/04/04/pdfs/BOE-A-2009-5688.pdf

"Pisos salariales mínimos – Unión informática" https://unioninformatica.org/institucional/convenio-colectivo-de-trabajo/

"La realidad del perfil de informático júnior en España según los informes – Xataka" https://www.xataka.com/tecnologiazen/la-realidad-del-perfil-de-informatico-junior-en-espana-segun-los-informes

"Precio neto de la electricitdad – Mincotur" https://www.mincotur.gob.es/es-es/IndicadoresyEstadisticas/BoletinEstadistico/IV.%20Energ%C3%ADa%20y%20emisiones/IV_12.pdf

"La oficina ideal – El Pais" https://cincodias.elpais.com/cincodias/2014/10/28/pyme/1414500383_553511.html

"Determinar el impacto de los riesgos en el costo del proyecto mediante el VME" https://guiadeproyecto.com/2015/01/12/determinar-el-impacto-de-los-riesgos-en-el-costo-del-proyecto-mediante-el-vme/

"12 técnicas para la estimación de costes en proyectos – Tendencias & Innovación, Marc Bara" https://www.obs-edu.com/es/blog-investigacion/project-management/12-tecnicas-para-la-estimacion-de-costes-en-proyectos

"Reservas de contingencia y reservas de gestión – Jose Huerta" https://josehuerta.es/gestion/proyectos/costes/reservas-de-contingencia-y-reservas-de-gestion

"Xcode para Windows con Smarface para desarrollo nativo de ios – Smartface" https://www.smartface.io/xcode-para-windows-con-smartface-para-desarrollo-nativo-de-ios/

"Mac rental service that provides a PC an Mopbile users remote Access ro our Mac servers through the Cloud – Macincloud" https://www.macincloud.com/

"Gestió de Projectes Software: Gestió àgil de Projectes – Documentación de la asignatura GPS, FIB"

"Especificació Requisits, Entrega final GPS – Silviu Chirvasa, Daniel Garrapucho, David Marin i Jonathan Nebot"

"Requeriments specification in agile projects: An oxímoron? – Albert Tort, Documentación de la asignatura ER, FIB"

"Especificación de Requisitos, Entrega final ER – Sara Bourjila, Sergi Casas, Manel Fernández, Adolfo López, David Marin"

"Editor interactivo online de diagramas – JGraph Ltd." https://www.draw.io/

"Patrones de Arquitectura MVC, MV-VM, MVP – Gilber Aranguren" https://ingsoftwarei2014.wordpress.com/category/comparacion-de-los-patrones-de-arquitectura-mvc-mv-vm-mvp/

"Model-View-ViewModel (MVVM) Explained – Jeremy Likness" https://www.wintellect.com/model-view-viewmodel-mvvm-explained/

"MVC y MVVM – Adictos al trabajo" https://www.adictosaltrabajo.com/2012/10/07/zk-mvc-mvvm/

"El patrón MVVM en Xamarin Forms – Software Crafters" https://softwarecrafters.io/xamarin/patron-mvvm-xamarin-forms/

"El patrón Model-View-ViewModel – Microsoft" https://docs.microsoft.com/es-es/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

"Parte 5. Enlaces de datos a MVVM – Microsoft" https://docs.microsoft.com/es-es/xamarin/xamarin-forms/xaml/xaml-basics/data-bindings-to-mvvm

"XAMARIN.FORMS Y MVVM – Bravent" https://www.bravent.net/xamarin-forms-y-mvvm

"Inicio rápido: Instalación y uso de un paquete en Visual Studio – Microsoft" https://docs.microsoft.com/es-es/nuget/quickstart/install-and-use-a-package-in-visual-studio

"Mobile view lifecycle. Fixing Xamarin.Forms Page lifecycle – Jlamch" https://jlamch.blogspot.com/2018/08/mobile-view-lifecycle-fixing.html

12. Anexo

12.1. Códigos externos

ACR User Dialogs

```
this.Open = new Command(() =>
{
    // var icon = await BitmapLoader.Current
        .LoadFromResource("emoji_cool_small.png", null, null);

    ToastConfig.DefaultBackgroundColor = System.Drawing.Color.AliceBlue;
    ToastConfig.DefaultMessageTextColor = System.Drawing.Color.Red;
    ToastConfig.DefaultActionTextColor = System.Drawing.Color.DarkRed;
    //var bgColor = FromHex(this.BackgroundColor);
    //var msgColor = FromHex(this.MessageTextColor);
    //var actionColor = FromHex(this.ActionTextColor);

    dialogs.Toast(new ToastConfig(this.Message)
        //.SetBackgroundColor(bgColor)
        //.SetMessageTextColor(msgColor)
        .SetDuration(TimeSpan.FromSeconds(this.SecondsDuration))
        .SetPosition(this.ShowOnTop ? ToastPosition.Top : ToastPosition.Bottom)
        //.SetIcon(icon)
        .SetAction(x => x
            .SetText(this.ActionText)
            //.SetTextColor(actionColor)
            .SetAction(() => dialogs.Alert("You clicked the primary toast button"))
        )
    );
});
```

Figura 15: Código de un Toast lanzado desde un Command.

Newtonsoft.Json

```
if(listaFiltros != null && listaFiltros.Count > 0)
    request.AddQueryParameter("FILTROS",
                        JsonConvert.SerializeObject(new ListaFiltros(listaFiltros)));

client.Timeout = AjustesApp.ServicioWebTimeOut;
var response = await client.ExecuteTaskAsync(request);

var result = response.Content;
if (!response.IsSuccessful)
    return null;

RespuestaWS<List<Espacio>> resultado =
                    JsonConvert.DeserializeObject<RespuestaWS<List<Espacio>>>(result);
if (resultado == null || resultado.Objeto == null)
    return null;

return resultado.Objeto;
```

Figura 16: Serliazación y deserialización de distintos objetos en JSon.

Permissions Plungin y Geolocator Plungin

```
try
{
    var status = await CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Location);
    if (status != PermissionStatus.Granted)
    {
        var results = await
                        CrossPermissions.Current.RequestPermissionsAsync(Permission.Location);
        //Best practice to always check that the key exists
        if(results.ContainsKey(Permission.Location))
        status = results[Permission.Location];
    }

    if (status == PermissionStatus.Granted)
    {
        var results = await CrossGeolocator.Current.GetPositionAsync(10000);
        LabelGeolocation.Text = "Lat: " + results.Latitude + " Long: " + results.Longitude;
    }
    else if(status != PermissionStatus.Unknown)
    {
        await DisplayAlert("Location Denied", "Can not continue, try again.", "OK");
    }
}
catch (Exception ex)
{
    LabelGeolocation.Text = "Error: " + ex;
}
```

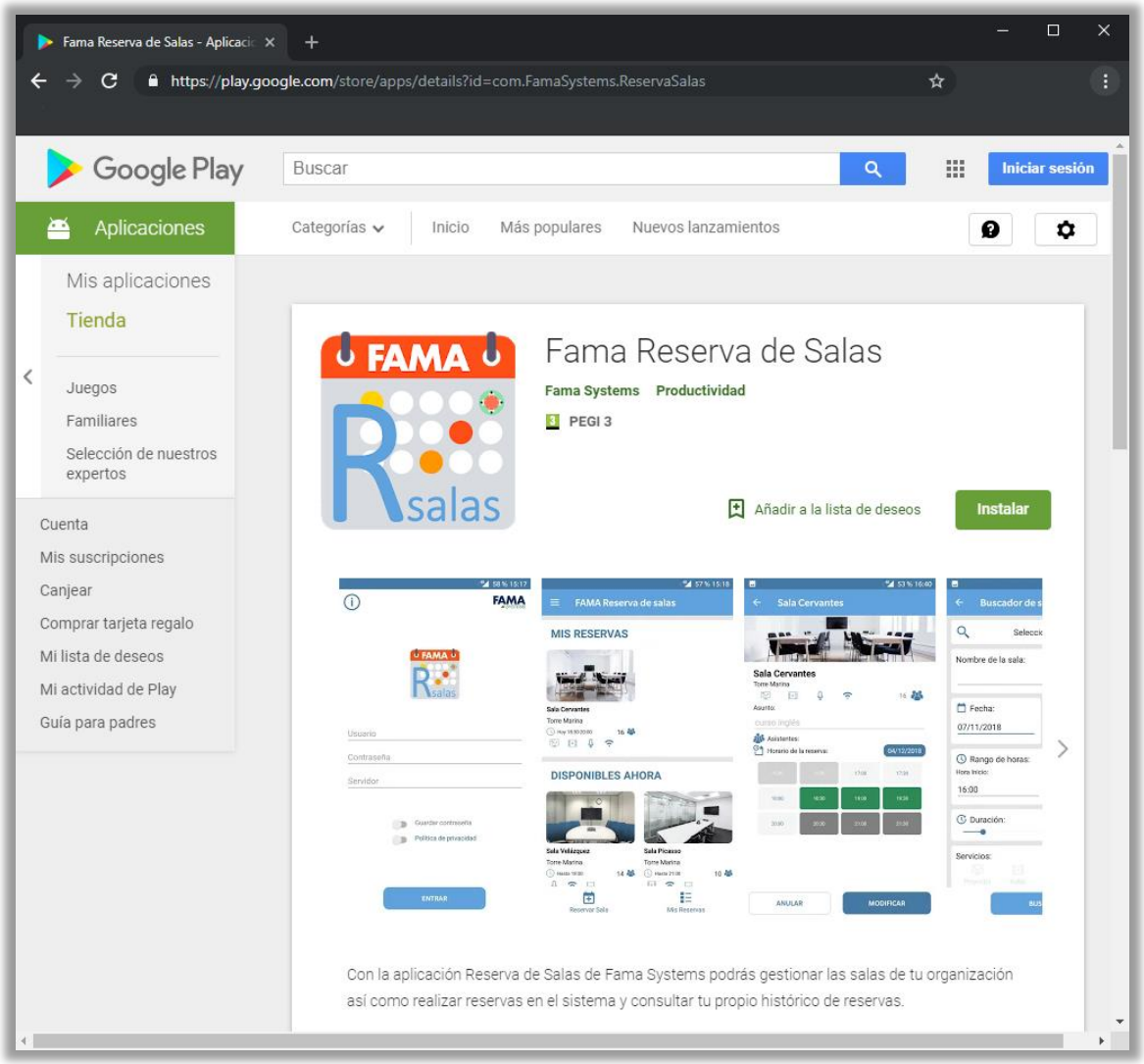
Figura 17: Comprobación de permisos para localización y obtención de la localización.

Media Plungin

```
takePhoto.Clicked += async (sender, args) =>
{
    await CrossMedia.Current.Initialize();

    if (!CrossMedia.Current.IsCameraAvailable || !CrossMedia.Current.IsTakePhotoSupported) {
        DisplayAlert("No Camera", ":( No camera available.", "OK");
        return;
    }

    var file = await CrossMedia.Current.TakePhotoAsync(
                                            new Plugin.Media.Abstractions.StoreCameraMediaOptions
    {
        Directory = "Sample",
        Name = "test.jpg"
    });

    if (file == null)
        return;

    await DisplayAlert("File Location", file.Path, "OK");

    image.Source = ImageSource.FromStream(() =>
    {
        var stream = file.GetStream();
        return stream;
    });
};
```

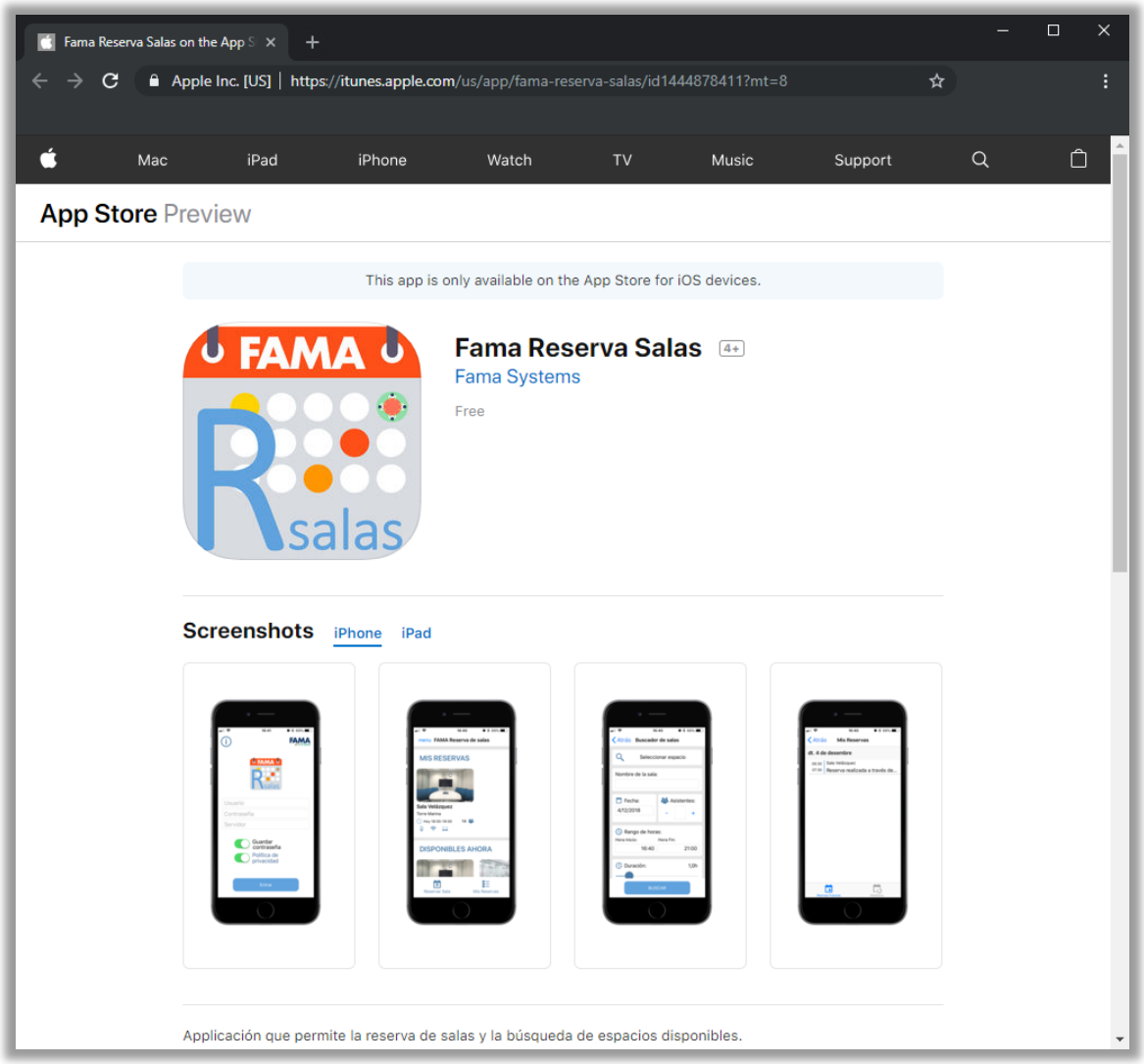
Figura 18: Obtención de una imagen desde la cámara del dispositivo.

12.2. Página de *Google Play* de la Aplicación



Enlace:
https://play.google.com/store/apps/details?id=com.FamaSystems.ReservaSalas

12.3. Página de *App Store* de la Aplicación



Notas: El icono de la aplicación en iOS no admite transparencias, por lo que en futuras versiones se deberá personalizar para este sistema.

Enlace: https://itunes.apple.com/us/app/fama-reserva-salas/id1444878411